

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Application of:

Myongsu Choe

Serial No.: *To be assigned*

Examiner: *To be assigned*

Filed: 20 December 2001

Art Unit: *To be assigned*

For: **APPARATUS AND METHOD FOR PERFORMING HIGH-SPEED IP
ROUTE LOOKUP AND MANAGING ROUTING/FORWARDING TABLES**



CLAIM OF PRIORITY UNDER 35 U.S.C. §119


The Assistant Commissioner
for Patents
Washington, DC 20231

Sir:

The benefit of the filing date of the following prior foreign applications, Korean Priority No. 2001-7568 (filed in Korea on 15 February 2001), and filed in the U.S. Patent and Trademark Office on 20 December 2001 and Provisional Application Serial No. 60/257,148 (filed in the U.S. on 22 December 2000) are hereby requested and the right of priority provided in 35 U.S.C. §119 is hereby claimed.

In support of this claim, filed herewith is a certified copy of said original foreign and U.S. applications.

Respectfully submitted,


Robert E. Bushnell
Reg. No.: 27,774
Attorney for the Applicant

1522 "K" Street, N.W., Suite 300
Washington, D.C. 20005-1245
(202) 408-9040

Folio: P56293
Date: 12/20/01
I.D.: REB/mn

1291736

J1073 U.S. PTO
10/022210



대한민국 특허청

KOREAN INTELLECTUAL PROPERTY OFFICE

별첨 사본은 아래 출원의 원본과 동일함을 증명함.

This is to certify that the following application annexed hereto
is a true copy from the records of the Korean Intellectual
Property Office.

출원번호 : 특허출원 2001년 제 7568 호
Application Number

출원년월일 : 2001년 02월 15일
Date of Application

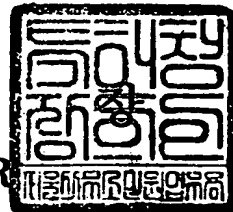
출원인 : 삼성전자 주식회사
Applicant(s)



2001 년 03 월 20 일

특 허 청

COMMISSIONER



CERTIFIED COPY OF
PRIORITY DOCUMENT

【서류명】	특허출원서
【권리구분】	특허
【수신처】	특허청장
【참조번호】	0003
【제출일자】	2001.02.15
【국제특허분류】	G06F
【발명의 명칭】	고속 인터넷프로토콜 라우터의 라우팅/포워딩 룩업 및 라우팅 테이블 관리
【발명의 영문명칭】	ROUTE LOOKUP AND ROUTING/FORWARDING TABLE MANAGEMENT FOR HIGH-SPEED INTERNET PROTOCOL ROUTER
【출원인】	
【명칭】	삼성전자 주식회사
【출원인코드】	1-1998-104271-3
【대리인】	
【성명】	이건주
【대리인코드】	9-1998-000339-8
【포괄위임등록번호】	1999-006038-0
【발명자】	
【성명의 국문표기】	최명수
【성명의 영문표기】	CHOE, Myong Su
【주민등록번호】	590822-1058410
【우편번호】	442-470
【주소】	경기도 수원시 팔달구 영통동 1044-3 203호
【국적】	KR
【우선권주장】	
【출원국명】	US
【출원종류】	특허
【출원번호】	60/
【출원일자】	2000.12.22
【증명서류】	미첨부
【취지】	특허법 제42조의 규정에 의하여 위와 같이 출원합니다. 대리인 주 (인) 이 건

【수수료】

【기본출원료】 20 면 29,000 원

【가산출원료】 38 면 38,000 원

【우선권주장료】 1 건 26,000 원

【심사청구료】 0 항 0 원

【합계】 93,000 원

【첨부서류】

1. 요약서·명세서(도면)_1통

【요약서】**【요약】**

본 발명은 무작위 추출 알고리즘인 스킵 리스트와 해시 테이블 등을 이용하여 인터넷프로토콜 어드레스의 고정 또는 가변의 프리픽스 범위에 따라 스킵 리스트상의 노드가 생성되며, 각 노드에서 각각 설정된 프리픽스 길이에 따른 해시 테이블 형태로 라우팅 엔트리가 저장되는 방식을 제공하여, 최소한의 경로검색시간 제공 및 관련 라우팅/포워딩 테이블들의 효율적인 관리를 통해 최단 경로검색이 가능하도록 한다.

【대표도】

도 6

【색인어】

IP 라우터, 라우팅 테이블, 포워딩 테이블, 룩업, Longest Prefix Matching, 무작위추출 알고리즘

【명세서】**【발명의 명칭】**

고속 인터넷프로토콜 라우터의 라우팅/포워딩 룩업 및 라우팅 테이블 관리{ROUTE LOOKUP AND ROUTING/FORWARDING TABLE MANAGEMENT FOR HIGH-SPEED INTERNET PROTOCOL ROUTER}

【도면의 간단한 설명】

도 1은 본 발명이 적용될 수 있는 라우터 구조의 일례로서, 분산구조를 갖는 라우터의 개략적인 구조도.

도 2는 본 발명이 적용될 수 있는 라우터 구조의 일례로서, 병렬구조를 갖는 라우터의 개략적인 구조도.

도 3은 본 발명이 적용될 수 있는 전송 엔진(Forwarding Engine)의 개략적인 구조도.

도 4는 본 발명이 적용되는 라우팅 프로세서의 개략적인 구조도.

도 5는 본 발명의 일 실시예에 따른 룩기스트 프리픽스 검색을 설명하기 위한 도면.

도 6은 본 발명의 일 실시예에 따른 스킵 리스트의 구조 예시도.

도 7a, 7b는 인터넷 상에서의 프리픽스 길이 분포에 대한 실측치를 나타낸 그래프.

도 8은 본 발명의 일 실시예에 따른 헤더 노드의 생성을 설명하기 위한 헤더 노드 구조 예시도.

도 9는 본 발명의 일 실시예에 따른 스킵 리스트의 전체 구축 작업을 설명하기 위한 구조 예시도.

도 10a, 10b는 본 발명의 일 실시예에 따라 프리픽스 길이에 따른 라우팅 테이블의 구축을 설명하기 위한 스킵 리스트 구조 예시도.

도 11은 본 발명의 일 실시예에 따른 라우팅 테이블 생성 과정의 개략적인 흐름도.

도 12는 본 발명의 일 실시예에 따른 라우팅 테이블 갱신 과정의 개략적인 흐름도.

도 13, 14는 본 발명의 일 실시예에 따른 라우트 검색 및 갱신 작업을 설명하기 위한 스킵 리스트 구조 예시도.

도 15는 본 발명의 일 실시예에 따른 라우트 룩업 진행 과정의 개략적인 흐름도.

도 16은 본 발명의 일 실시예에 따른 전체 룩업 작업 과정을 설명하기 위한 스킵 리스트 구조도.

도 17은 본 발명의 일 실시예에 따른 각 해시 테이블에서의 목적지 IP 어드레스 감소 방식을 설명하기 위한 도면.

【발명의 상세한 설명】

【발명의 목적】

【발명이 속하는 기술분야 및 그 분야의 종래기술】

<17> 본 발명은 인터넷 상에서 패킷(packet)을 해당 목적지로 전송하기 위한 라우팅(routing) 기술에 관한 것으로, 특히 고속으로 IP(Internet Protocol) 라우팅 룩업(routing lookup), 및 라우팅 테이블 관리에 관한 것이다.

<18> 현재 인터넷 이용자의 급증과, 제공되는 서비스의 다양화 및 VoIP(Voice over Internet Protocol) 등의 이용분야 확대 등에 따라 인터넷 상에 기하급수적인 트래픽 폭증이 초래되고 있다. 이에 따라 인터넷상에서 패킷을 해당 목적지로 단시간에 전송하기 위해, 특히 수십 기가(Giga) 또는 테라(Tera)급 고속 라우터(router)의 경우, 해당 패킷의 목적경로를 지연 없이 최단 시간 내에 발견하여 포워딩(forwarding)해야 하는 것이 기술적 현안으로 대두되었다. 이를 위해서 라우터 각각의 물리적 입력 인터페이스를 통해 전달되는 패킷의 목적지 경로를 찾아내기 위해 최단 경로(라우팅 또는 포워딩) 테이블의 유지, 효율적인 경로테이블 관리 및 검색시간 단축 등의 기능이 필수적으로 요구되고 있다.

<19> 초창기의 라우터에서는 도달한 패킷의 목적지를 검색 및 처리하는데 요구되는 시간이 라우터에 접속된 전송로를 통해 도달하는 패킷의 전송시간에 비해 고속이어서, 인터넷상에서 각각의 서브네트워크 사이를 연결하는 접속 노드로서의 라우터의 고유기능 수행에는 큰 어려움이 없어 왔다. 반면에 최근 들어 고속의 광 네트워크 인터페이스의 대역폭 증가와 더불어(예를 들어, POS OC-192(10 Gbps) 또는 IP over DWDM) 해당 전송로상의 패킷의 전송시간이 라우터 내에서 패킷의 검색 및 이의 처리시간에 비해 상대적으로 짧아지게 되었다. 특히 수백 기가 또는 수 테라급의 백본(backbone)급 라우터의 경우에 있어서는 라우터내에서 패킷 경로 검색으로 인해 소요되는 시간 등으로 인해 라우터가 초고속 인터넷상에서 병목(bottleneck)의 주요 원인을 제공한다는 비난을 받아 왔다. 이러한 기술로는 본 명세서의 말미에 첨부되는 참조 문헌 리스트 중에서 1, 2, 3, 4, 5번으로 표시된 문헌들에 개시된 바를 예로 들 수 있다.

<20> 이에 따라 1990년대 초반에 인터넷의 기술표준을 관리하는 IETF(Internet

Engineering Task Force)에서는, IPv4(IP version4)에서의 IP주소를 효율적으로 사용하는 CIDR(Classless Inter-Domain Routing) 방식이 워킹 그룹(working group)에 의해 표준으로 제안되어 적용되어 왔다. 이는 하기 첨부되는 참조문헌 리스트 중 7, 8번으로 표시된 문헌들에 상세히 개시되어 있다.

<21> 이 경우에 다양한 프리픽스(Prefix) 길이를 가지는 패킷에서 해당 프리픽스를 파악하기 위하여, 롱기스트 프리픽스(Longest Prefix)에 일치(matching)하는 IP주소를 발견하기 위해 일반적으로 트라이(trie)(또는 Patricia trie) 데이터 구조를 기반으로 하는 롱기스트 프리픽스 매칭(Longest Prefix Matching: LPM) 검색을 사용하여 왔다. 이러한 롱기스트 프리픽스 매칭 검색 방식은 하기 첨부되는 참조 문헌 리스트 중 9번으로 표시되어 있는 문헌에 개시된 바를 예로 들 수 있다.

<22> 이하, 하기 표 1을 참조하여 롱기스트 프리픽스 매칭 검색 방식에 대해 상세히 설명하기로 한다. 하기 표 1은 라우팅(또는 포워딩) 테이블에 포함되는 라우팅 엔트리의 간단한 예를 나타낸다. IPv4의 경우에는 프리픽스가 원래 32를 초과할 수 없지만, 하기 표 1에서 별표(*)는 '0' 또는 '1'의 숫자로 이루어진 임의의 숫자들의 삽입구를 의미한다. 예를 들어 '0*'은 '01*' 또는 '011*' 또는 '010*' 등을 포함하며 나머지 비트 스트링이 어떠한 것인지는 문제되지 않는다.

<23>

【표 1】

Prefix	Interface
0*	00
01010*	10
010101*	10
0101011*	11
1*	11
11*	01
10*	11
1110*	01
100101011*	11
100101010*	11
1001010101*	10

<24> 라우팅 엔트리 또는 라우트는 '<network prefix>,<host number>'로 이루어진 IP 어드레스로 표현된다. 상기 표 1을 참조하여 목적지 IP 어드레스가 '10010101000...0'일 경우에 대해 언급해 보면, 이러한 목적지 IP 어드레스의 첫 번째 비트는 하기 라우팅 테이블에 기재된 네트워크 프리픽스(이하, 간단히 '프리픽스'라 칭함)와 비교된다. 이에 따라 '0*', '01010*', '010101*' 및 '0101011*'은 자신의 첫 번째 비트가 '0'으로 시작되기 때문에 제외된다. 이후 주어진 목적지 어드레스와 나머지 프리픽스 어드레스를 비교할 경우에, 프리픽스 '11*', '1110*', 및 '1001010101*'이 상기 목적지 어드레스와 나머지 두 번째로 또는 10번째 비트에서 일치되지 않기 때문에 제외된다. 이후 나머지 대상 프리픽스(즉, '1*', '10*', '100101010*', 및 '1001010101*') 중에서, '1001010101*'이 롱기스트 프리픽스(또는 most specific) 일치 라우트로 선택되고, 해당 목적지 IP 어드레스를 갖는 패킷은 인터페이스 '10'을 통해 인접 라우터로 전송된다.

<25> 한편, 현재 발표된 경로 검색 알고리즘은 메모리 액세스에 걸리는 시간을 단축하기 위해, 가능한 해당 프로세서(라우팅 프로세서 또는 포워딩 엔진)내의 캐시 크기 내에

저장 및 사용될 수 있도록 하여 효율적인 검색 및 최소한의 메모리 액세스를 요구하기 위한 연구가 주로 이루어져 왔다.

<26> 그런데, 상기한 라우팅 테이블에서의 경로검색 알고리즘은 효율적인 검색을 지원하지만 라우팅 테이블 또는 포워딩 테이블(Forwarding Table)의 변경은 거의 불가능하도록 비효율적으로 설계되어 변경된 경로의 반영에 따른 지연시간을 초래할 수 있다.

<27> 더구나, 라우팅 프로세서(120)에서의 라우팅 테이블은 DMA(Direct Memory Access) 또는 IPC(Inter-Processor Communication) 카드 또는 스위치 패브릭(130) 등을 통해 복사되어 포워딩 엔진의 포워딩 테이블에 적용되는데, 이때 해당 변경된 경로만의 추가 및 삭제가 아닌 새로이 포워딩 테이블을 구축해야 한다. 이에 따른 추가적인 시간 지연이 초래되며, 또한 라우터의 메모리 액세스 요구에 따른 메모리 대역폭 사용 증가 등에 따라 라우터의 내부 메모리 또는 시스템버스 상에 병목을 초래할 수 있다.

<28> 더구나, 일부 알고리즘의 경우에는 최초 라우팅 테이블 (또는 포워딩 테이블) 구성 시에 현재 라우터의 연결상태를 보여주는 'link reachability information'을 사용하여 해당 테이블을 구성하여야 한다. 따라서, 해당 연결정보를 순차적으로 라우팅 테이블에 삽입하기 위해서는 사전에 경로의 프리픽스 길이별로 모든 경로들을 분류(sorting)하는 추가 선행작업이 필요하다. 이러한 알고리즘의 예로는 하기 첨부되는 참조문헌 리스트 중 10, 11, 12, 14, 15번으로 제시한 문헌을 예로 들 수 있다.

<29> 종래 대표적으로 사용하는 데이터 구조인 'Radix tree' 또는 'PATRICIA trie'는 패킷 경로를 찾기 위해 요구되는 메모리 액세스수가 증가할 뿐만 아니라 해당 라우터 주변에서 인접 라우터들로부터 경로 설정 또는 삭제에 의한 경로변경을 반영하기 위해서는 상당한 갱신(update)시간이 요구된다.

<30> 10번의 참조 문헌에 제시된 바와 같이, 스웨덴의 'Lulea university'에서는 포워딩 엔진에서 캐쉬 크기 내에 저장 가능한 컴팩트한 포워딩 테이블을 제시하였지만, 한편으로 해당 테이블의 경로변경을 반영하기는 곤란하다. 11번 문헌에 제시된 바와 같이, 'Srinivasan et al.'이 제시한 'controlled prefix expansion'을 위한 방식 (이와 관련하여 미국특허 제6,011,795호가 George Varghese와 Srinivasan Venkatachary에게 허여되어 있음) 및 13번 참조 문헌에 제시된 바와 같이, 'Tzeng et al.'이 제안한 알고리즘도 크게는 'multi-resolution trie'에 근간을 두는 방식이지만, 트라이의 자료구조 속성상 경로의 추가 및 삭제가 용이하지 않다. 이와 관련하여 미국특허 제6,061,712호 및 제6,067,574호가 Hong-Yi Tzeng에게 허여되어 있다. 한편, 상기 14번, 15번 문헌에 제시된 바와 같이, 'Waldvogel et al.'이 제시한 'rope search'알고리즘도 기본적으로 트라이를 기반으로 해싱(hashing)을 사용하는 방식이지만 변경된 경로에 대한 갱신측면에서는 트라이와 같이 비효율적임을 예상할 수 있다. 이와 관련하여서 미국특허 제6,018,524호가 Jonathan Turner, George Varghese, 및 Marcel Waldvogel에게 허여되어 있다.

<31> 이외에도, 트라이내에서 검색경로를 줄이기 위한 몇 가지 방안 (예를 들면, 하기 참조 문헌 리스트 중 16번 문헌에 제시된 바와 같이, 'two trie'를 이용한 'route lookups'와, 하기 17번 문헌에 제시된 바와 같이, 트라이의 레벨 길이를 단축하기 위한 방안 및 9번 문헌의 'DP-트라이'도 제안되어 어느정도 검색시간의 단축을 가져왔지만, 해당 라우팅 테이블 (또는 포워딩 테이블)의 경로 변경의 반영에는 여전히 문제점을 갖고 있다.

<32> 또한, 하드웨어적으로 처리하기 위한 방법도 일부 검토되었는데 그 중에서 하기 19번 문헌에 개시된 바와 같은, 대용량의 DRAM 사용을 통한 방법이 제시되었다. 이러한 방식은 검색시간 자체는 기존의 소프트웨어적으로 구현된 방안에 비해 빠르지만, 현재의

IPv4를 대체하기 위해 개발중인 128비트의 어드레스를 갖는 IPv6(Internet Protocol version 6)으로 천이 할 때에는 해당 메모리소자의 용량 또는 가격 등의 요소로 인해 현실적으로 구현이 불가능한 결점이 존재한다. 또한, 하기 20번 문헌에 개시된 바와 같은 CAM(Content Addressable Memory)을 사용하여 하드웨어적으로 구현하기 위한 방법도 제시되었지만, 비용 등의 문제가 있어 현재는 고려되지 않고 있다. 하기 21번 문헌에 개시된 바와 같은, 'Huang et al.'은 메모리 액세스빈도를 줄이기 위해 파이프라인(pipeline)식으로 메모리를 액세스 할 수 있는 'indirect lookup'알고리즘을 제시하였으나, 이 역시 IPv6에 적용 곤란한 결점을 갖고 있다.

【발명이 이루고자 하는 기술적 과제】

- <33> 따라서 본 발명의 목적은 무작위추출(randomized) 알고리즘을 사용하여 최소한의 메모리 액세스 시간을 가지며 메모리의 경제적 사용을 허용하는 고속 IP 라우터의 라우팅 룩업과 라우팅(및 포워딩) 테이블 관리 방법을 제공함에 있다.
- <34> 본 발명의 다른 목적은 경로의 추가 및 삭제에 대한 변경 및 관리가 용이한 고속 IP라우터의 라우팅 룩업과 라우팅(및 포워딩) 테이블 관리 방법을 제공함에 있다.
- <35> 본 발명의 또 다른 목적은 해당 테이블 구성시 입력하고자 하는 경로의 프리픽스 길이에 무관하게, 즉 프리픽스 길이에 따라 관련 라우트들의 분류(sorting)없이 라우팅 테이블과 포워딩 테이블을 효율적으로 구축할 수 있는 고속 IP 라우터의 라우팅 룩업과 라우팅(및 포워딩) 테이블 관리 방법을 제공함에 있다.
- <36> 상기한 목적을 달성하기 위하여 본 발명의 제1 특징에 따르면, 프리픽스 길이 범위

를 키로 하는 인터넷프로토콜 어드레스 검색을 방법에 있어서, 미리 설정된 최대 값의 키를 갖는 헤더 노드와, 감소되는 순서로 미리 설정된 고정된(또는 가변된) 범위의 키를 각각 가지며, 각 프리픽스 길이에 해당하는 각각의 해시 테이블이 구비되어 해시 테이블에 각각의 프리픽스 길이에 해당하는 라우트 엔트리가 저장되는 다수의 노드를 포함하여 구성되는 스킵 리스트를 이용함을 특징으로 한다.

<37> 본 발명의 제2 특징에 따르면, 스킵 리스트를 이용한 인터넷프로토콜 라우팅 테이블 생성 방법에 있어서, 스킵 리스트내의 모든 노드를 다루기 위해 최대 레벨을 가진 헤더 노드를 생성하는 과정과, 미리 설정된 분할의 프리픽스 범위를 가진 다수의 노드를 상기 스킵 리스트에 삽입하는 과정과, 프리픽스 범위에 해당하는 각 노드에서 각각의 주어진 라우트 엔트리를 위한 해시 테이블이 생성하는 과정을 가짐을 특징으로 한다.

<38> 본 발명의 제3특징에 따르면 인터넷프로토콜 어드레스의 프리픽스 범위에 따라 노드가 생성되며, 각 노드에서 각각 설정된 프리픽스 길이에 따른 해시 테이블 형태로 라우팅 엔트리가 저장되는 스킵 리스트를 이용한 라우팅 테이블의 갱신 방법에 있어서, 갱신할 라우트의 프리픽스 길이에 해당하는 프리픽스 범위가 설정된 노드를 찾는 과정과, 찾은 노드에서 갱신할 라우트와 동일한 프리픽스 길이를 가진 해시 테이블을 검색하는 과정과, 해당 해시 테이블이 발견시에 해당 해시 테이블에서 해당 라우트를 갱신하는 과정을 가짐을 특징으로 한다.

<39> 본 발명의 제4특징에 따르면, 인터넷프로토콜 어드레스의 프리픽스 범위에 따라 노드가 생성되며, 각 노드에서 각각 설정된 프리픽스 길이에 따른 해시 테이블 형태로 라우팅 엔트리가 저장되는 스킵 리스트를 이용한 라우팅 테이블의 라우트 록업에 있어서, 스킵 리스트의 제1노드에서부터 시작하여 인접한 노드로 찾아가는 과정과, 해당 노드에

서 각각 목적지 어드레스와 각각의 해시 테이블이 프리픽스 길이가 감소되는 순서로 비교하는 과정과, 비교 결과 상기 비교하는 해시 테이블이 상기 목적지 어드레스를 포함할 경우에 일치하는 프리픽스를 룽기스트 프리픽스로 간주하는 과정을 가짐을 특징으로 한다.

<40> 본 발명의 위와 같은 그리고 다른 목적, 특징, 및 장점 등은 첨부된 도면을 참조한 다음의 상세한 설명으로부터 명백해질 것이다.

【발명의 구성 및 작용】

<41> 이하 본 발명에 따른 바람직한 실시예를 첨부한 도면을 참조하여 상세히 설명한다. 하기 설명에서는 구체적인 구성 소자 등과 같은 특정 사항들이 나타나고 있으나, 이는 본 발명에 대한 이해를 돕기 위해서 제공된 것일 뿐 이러한 특정 사항들이 본 발명의 범위 내에서 소정의 변형이나 혹은 변경이 이루어질 수 있음은 이 기술분야에서 통상의 지식을 가진 자에게는 자명할 것이다.

<42> 먼저 본 발명이 적용될 수 있는 라우터 및 라우팅 방식을 설명하기로 한다. 3번 참조 문헌에는 고속 라우터 구조의 최근 주요 경향으로서 분산구조와 병렬 구조의 라우터에 대해 분류하고 있다. 이러한 두 구조는 포워딩 엔진의 위치에 따라 구분된다. 현재 시장에 출하되는 백본급의 고속 라우터는 효율적인 패킷처리를 위해서 과거의 집중구조에서, 도 1에서 보는 바와 같이, 분산구조를 취하는 추세이다.

<43> 도 1은 본 발명이 적용될 수 있는 일례로서, 분산구조를 갖는 라우터의 개략적인 구조도이다. 도 1을 참조하면, 라우터는 패킷이 입력 또는 출력되는 포워딩 엔진(FE: Forwarding Engine)이 장착된 라인 카드 모듈(line card module)(110)과, 초기 라우팅

테이블(RT) 구축 및 라우팅 테이블을 관리하는 라우팅 프로세서(routing processor)(120) 및 패킷이 라우터 내에서 특정 포트(port)로 스위칭될 때에 사용되는 스위치 패브릭(switch fabric)(130)으로 구성된다.

<44> 상기 라우팅 프로세서(120)는 최근의 경로 변경이 반영된 라우팅 테이블(RT)을 구비한다. 라인 카드 모듈(110)에서 각 포워딩 엔진(FE)은 라우팅 프로세서(120)의 라우팅 프로토콜(RIP, OSPF, 또는 BGP4등)로부터 생성 및 관리된 라우팅 테이블 중 일부 영역(field)이 복사되어 효율적인 검색을 위해 특별히 고안된 포워딩 테이블(FT: Forwarding Table)을 구비한다. 이러한 전송 테이블(FT)은 검색 효율만을 강조하기 위해 특별히 설계된 데이터 구조를 갖고 있으며, 반면에 경로의 추가 또는 삭제의 경우에는 비교적 비효율적인 구조를 갖고 있다.

<45> 만일 라인 카드 모듈(110)에 입력되는 패킷이 포워딩 테이블(FT)에서 목적경로를 발견할 수 없으면, 해당 패킷은 스위치 패브릭(130)을 통하여 라우팅 프로세서(120)로 이동하여 이 곳에서 라우팅 테이블(RT)을 검색하여 최종 목적지로 가기 위한 인접 라우터 인터페이스를 발견한다. 이후 이동된 패킷은 해당 인터페이스로 전송하기 위해 스위치 패브릭(130) 및 출력 라인 카드 모듈(110)을 통해 전송되어진다. 만일 상기 라우팅 프로세서(120)에서 라우팅 테이블(RT)을 통해서도 해당되는 목적 경로를 발견할 수 없을 경우에는 해당 패킷은 라우팅 프로세서(120)에서 버려진다.

<46> 한편, 첨부 참조 문헌 리스트 중 6번 문헌(Bell Labs) 및 4번 문헌(BBN)에는 병렬 구조의 라우터에 대해 개시하고 있다. 도 2는 본 발명이 적용될 수 있는 일례로서, 병렬 구조를 갖는 라우터의 개략적인 구조도이다. 도 2에 도시된 바와 같은 병렬 구조를 갖는 라우터에서는 포워딩 테이블(FT)을 구비한 각 포워딩 엔진(112)이 라인 카드 모듈(110)

에서 별도 분리되는 구조를 가진다. 이러한 병렬 구조는 각 포워딩 엔진(112)에서 병렬 패킷 처리 및 라우트 룩업을 다루기 위해 클라이언트 및 서버 기반의 모델을 이용한다.

<47> 이러한 라우팅 프로세서(120)에서의 라우팅 테이블은 최근의 경로 변경에 대한 결과의 즉각적인 반영이 이루어질 수 있도록, 또한 유지 및 관리에 용이도록 설계되어야 한다. 또한 추가 또는 삭제되는 경로의 경우 포워딩 엔진(FE)에 있는 포워딩 테이블(FT)에 최소한의 비용으로 가능한 신속히 반영되어야 한다. 반영이 안될 경우에는 해당 수신(incoming) 패킷은 스위치 패브릭(130)을 통하여 라우팅 프로세서(120)에 보내짐으로써, 해당 패킷을 처리하기 위해 요구되는 추가적인 경로를 통한 바이패싱(bypassing)에 의해 전송 지연이 증대된다.

<48> 이상 본 발명이 적용될 수 있는 라우터 구조에 대해 간단히 언급하였으나, 본 발명이 적용될 수 있는 라우터 구조는 상기 열거된 구조로서 제한되는 것은 아니며, 여타의 다른 구조를 갖는 라우터에도 적용될 수 있음은 당업자에게 자명하다 할 것이다.

<49> 도 3은 상기 기술한 바와 같은, 분산 또는 병렬 구조를 갖는 라우터에서의 포워딩 엔진(예를 들어, 112)의 구조를 개략적으로 나타내며, 도 4는 본 발명이 적용되는 라우팅 프로세서(120)의 개략적인 구조를 나타낸다. 먼저 도 3을 참조하면 포워딩 엔진(112)은 다수의 입력 인터페이스(input interfacer1, 2...n)와 다수의 출력 인터페이스(output interfacer1, 2...m)를 가지며, 입력 인터페이스로 입력되는 패킷을 버퍼링하는 다수의 버퍼(buffer)와, 버퍼에 저장된 패킷을 등급별로 또는 종류별로 분류하는 패킷 분류기(packet classifier)와, 포워딩 테이블 및 상기 패킷 분류기에서 분류되어 출력되는 패킷을 상기 포워딩 테이블을 참조하여 해당 출력 인터페이스로 내보내는 라우트 룩업(route lookup) 제어부를 구비한다.

<50> 도 4를 참조하면, 라우팅 프로세서(120)는 상기 도 1 및 도 2에 도시된 바와 같은 스위치 패브릭(130)으로(부터) 송수신되는 패킷을 위한 다수의 입/출력 인터페이스(input/output interface1, 2...n)와, 다수의 입/출력 인터페이스의 패킷을 버퍼링하며 또한 스위치 패브릭(130)과 후단의 라우트 룩업 및 관리 제어부간을 인터페이스하는 스위치 인터페이스와, 라우팅 테이블 및 상기 스위치 인터페이스로부터 패킷을 제공받아 상기 라우팅 테이블을 참조하여 해당 입/출력 인터페이스로 내보내도록 상기 스위치 인터페이스에 제공하는 라우트 룩업 및 관리 제어부를 구비한다.

<51> 한편, 라우팅 테이블 또는 전송 테이블은 검색을 위해 주로 사용되는 자료구조인 배열(array), 트리(tree) 또는 트라이(trie) 등과 이를 다루기 위해 어떠한 알고리즘을 사용하는 것에 대해서는 원칙적으로 표준 관련 문서인 RFCs에서는 제약이 없다. 다만 룩업 프리픽스 매칭을 위해 최소한의 메모리 액세스 및 라우팅 테이블(또는 포워딩 테이블) 내에서 변경된 라우트의 수정이 용이해야 되는 현실성이 감안되어야 한다.

<52> 본 발명에서는 이러한 현실성을 감안하여 무작위추출 알고리즘(randomized algorithm)을 이용한다. 무작위추출 알고리즘은 실행 중에 임의선택을 하는 알고리즘이다. 무작위추출 알고리즘은 거의 모든 입력에 대해 잘 동작(well-behaved) 할 수 있는 알고리즘 설계 분야에 최근 주로 사용되고 있다. 이는 어떠한 입력에 대해 특정 가정을 근거로 처리하는 것이 아니라 (예를 들면, 동전을 던지는 것과 같은) 확률적인 선택에 따른다. 무작위추출 알고리즘의 장점은 그의 단순성과 효율성에 있다.

<53> 이러한 무작위추출 알고리즘에 기반을 둔 본 발명의 알고리즘은 최소한의 메모리 액세스와 라우팅 또는 포워딩 테이블의 손쉬운 관리를 제공한다. 라우트 엔트리 관리의 경우에는 종래의 전체 테이블의 재구성 방식 대신에 단지 변경된 라우트의 정보만 재구

성될 수 있도록 한다. 또한 종래의 일반적인 트리 또는 이진 검색 트라이와 같은 라우트
록업 알고리즘과는 달리 본 발명의 알고리즘은 도 5에 도시된 바와 같이 목적지 IP 어드
레스의 테일(tail)에서부터 롱기스트 프리픽스를 검색하게 된다.

<54> 또한 본 발명에서는 라우팅 테이블(또는 포워딩 테이블)을 위해 하기 참조문헌 리
스트 중 22번의 문헌으로 개시한 바와 같은 'W. Pugh'에 의해 제시된 스킵 리스트(skip
list)를 이용한다.

<55> 스킵 리스트(skip list)는 분류된(sorted) 노드를 갖는 링크 리스트(linked list)
로 간주되며 'balanced tree'를 대신하는 자료구조의 용도에 사용할 수 있다. 따라서, 스
킵 리스트는 종래의 대표적인 'balanced tree'(예를 들면, 하기 참조문헌 리스트 중 23번
문헌에 개시된 바와 같은 'Sleator'와 'Tarjan'에 의해 제시된 'Splay tree')에 비해 훨씬
단순하고 그리고 효율적으로 해당 데이터 구조상에 입력 및 삭제를 위해 사용될 수 있다

<56> 본 발명에서는 상기와 같은 스킵 리스트가 갖고 있는 장점상에 롱기스트 프리픽스
매칭(longest prefix matching)에 적합한 구조를 갖도록 변형시킨 스킵 리스트(variant
of skip list)를 제안한다.

<57> 도 6은 본 발명의 일 실시예에 따른 변형 스킵 리스트의 구조도이다. 도 6을 참조
하면, 각 노드(node)는 하나 또는 다수의 포인터를 가질 수 있으며, 왼쪽 첫번째의 노드
를 특히 헤더 노드(header node)(210)라 칭한다. 검색(searching), 추가, 삭제의 연산
(operation)은 항상 헤더 노드(210)에서부터 시작된다.

<58> 도 6을 참조하면, 헤더 노드(210)의 키(key)는 $+\infty$ 이며 각 노드(220, 230, 240,

250, 260)의 키는 감소 순서로 분류(sorting)됨을 알 수 있다. 각 키는 프리픽스 길이 범위(prefix length range) 또는 클러스터링(clustering)으로 그룹화되었음을 의미한다. 이는 기존의 트라이 또는 트리 구조와는 전혀 다른 방식으로서, 본 발명에서는 먼저 잎(leaves)에서부터 줄기(parent)로 검색한다.

<59> 두 번째 노드(220)의 키는 32~24의 프리픽스 길이 범위를 갖고 있으며 해당 범위에 속하는 프리픽스 길이를 갖는 라우트는 각 프리픽스 길이에 해당하는 해시 테이블(hash table)을 갖고 있다. 도 6을 참조하면, 두 번째 노드(220)의 범위에 속하는 라우트의 해시 테이블의 예로서 프리픽스 길이가 30비트인 해시 테이블(222)과 프리픽스 길이가 24비트인 해시 테이블(224)이 도시되고 있다. 또한 17~12의 프리픽스 길이 범위를 갖는 네 번째 노드(240)의 범위에 속하는 라우트의 해시 테이블의 예로서 프리픽스 길이가 15비트인 해시 테이블(242)이 도시되고 있다.

<60> 도 6에 도시된 바와 같이 본 발명의 실시예에서 프리픽스 길이는 임의로 균등히 나누었으나, 이외에도 가변의 길이를 갖는 노드의 그룹(group)으로 나눌 수 있다. 상기 Srinivasan과 Varghes'의 참조 문헌 11번에서는 메모리의 절약을 위해서 최적 프리픽스 범위를 위한 동적 프로그래밍(dynamic programming)을 사용하였다. 본 발명의 실시예에서도 이와 유사한 방식을 사용하여 프리픽스 길이 그룹을 나눌 수 있다. 그런데, 상기 도 6에 도시된 바와 같이 본 발명의 실시예에서는 미리 설정된 고정된 범위의 프리픽스 길이를 갖는 것이 바람직하다. 왜냐 하면 동적 프로그래밍에 의한 추가적인 계산 작업 로드를 피할 수 있기 때문이다.

<61> 예를 들어, IPv4에서 프리픽스 길이 범위가 8인 경우에는, 헤더 노드를 포함하여 5(이는 $1 + 32/8$ 로 유도됨)의 노드로 구성된다. 실제 프리픽스 길이의 비대칭적인 분산

(skewed distribution)은 특정 프리픽스 길이에서 빈번히 참조되는 국소성 패턴 (locality pattern)을 보여 주기 때문에, 이보다 실제로는 적은 수의 노드를 사용하는 것도 가능할 수 있다. 예를 들어 도 7a, 7b를 참조하면 프리픽스 길이가 약 17에서 27인 경우가 가장 빈번히 참조되는 것으로 나타나는데, 이에 따라 해당 프리픽스 길이를 포함하는 범위의 프리픽스 길이 키 값을 가지는 노드를 하나로 설정하고 다른 부분의 프리픽스 길이 범위에 따른 노드를 적절히 다수개 설정하여 구성할 수 있다.

<62> 다시 도 6을 참조하면, 헤더 노드(210)의 변수 'MaxLevel'은 스킵 리스트에 속한 모든 노드 중에서 최대 레벨값, 즉 헤더 노드에서 다른 노드를 포인팅하기 위한 총 포인터의 수를 나타낸다. 각 노드는 키 값인 'x->key'와 포인터 'x->forward[L](x 노드로부터 이웃한 Level-L 노드의 포인터)'를 간직하고 있다.

<63> 이때, 본 발명의 특징에 따라 상기 키 값은 프리픽스 길이 범위 또는 집합(set)의 값이 된다. 도 6의 예에서, 헤더 노드(210) 후단의 첫 번째 노드(220)에서는 32부터 24까지의 프리픽스 길이에 해당하는 범위의 라우트 엔트리를 가지며 해당 노드에서 각 프리픽스 길이는 각각의 해시 테이블을 갖는다. 각각의 해시 테이블은 해당 프리픽스에 일치하는 라우트 엔트리들을 저장한다. 만일 라우터에 저장하고자 하는 32~24범위의 모든 프리픽스 길이가 존재한다면 해당 노드는 32에서 24까지 9개의 해시 테이블을 가리키는 9개의 포인터를 가져야 한다. 각각의 해시 테이블은 단지 각 프리픽스 길이에 정확히 일치하는 라우트 엔트리를 저장한다.

<64> 이러한 스킵 리스트를 이용하는 본 발명의 특징에 따른 알고리즘에 대한 이해를 돕기 위하여, 이하 첨부 도면을 참조하여 헤더 노드의 생성 및 스킵 리스트의 구축에 대하여 보다 상세히 설명하기로 한다.

<65> 도 8은 본 발명의 일 실시예에 따른 헤더 노드의 생성을 설명하기 위한 도면이다.

도 8을 참조하면, 헤더 노드는 $+\infty$ 키 값과 최대 레벨 값 'MaxLevel'이 5이다. 헤더 노드는 포괄적으로 스킵 리스트 내의 모든 노드에 대해 다루어야 하기 때문에, IPv4인 경우에는, (하기 기술되는 확률 설정값 $p=1/2$ 인 경우) 2^{32} 에 달하는 라우트를 가진 데이터 구조용으로 'MaxLevel = 32'가 적절하다. 헤더 노드는 MaxLevel개의 전송 포인터를 가진다.

<66> 본 발명의 실시예에서는 프리픽스 길이 범위를 고정 또는 가변적으로 그룹화하여 클러스터링 하는 방식이 제안되며, 이에 따라 예를 들어, 총 프리픽스 길이가 32일 경우 고정 프리픽스 범위가 8인 것으로 가정할 경우에 MaxLevel이 4인 헤더노드가 요구된다

<67> 이와 같이, 헤더 노드가 생성되며, 이후 스킵 리스트의 다른 부분에 대한 구축을 도 9를 참조하여 설명하기로 한다. 도 9는 본 발명의 일 실시예에 따른 스킵 리스트의 전체 구축 작업을 설명하기 위한 구조 예시도이다. 먼저 만약 헤더 노드의 최 상위 레벨의 포인터가 'null'이면, 즉 헤더 노드의 포인터에 연결된 인접한 노드가 존재하지 않은 경우에 해당되며, 레벨은 레벨 번호가 '0'이 될 때까지 점점 감소된다. 도 9의 (a)를 참조하면, 프리픽스 길이 범위 값이 예를 들어, 32~24로 설정된 제1노드가 생성된 후 헤더 노드에서 레벨0의 포인터는 제1노드를 가리킨다. 이후 도 9의 (b) 및 (c)에 도시된 바와 같이 프리픽스 길이 범위가 23~18인 제2노드 및 프리픽스 길이 범위가 17~12인 제3노드가 차례로 생성된다. 이러한 생성과정에 의해 모든 프리픽스 길이 범위를 커버할 수 있도록 다수의 노드가 생성된다.

<68> 이때 생성된 상기 각각의 노드의 레벨은 프로그램 3에서 보여 주는 바와 같이 본

발명의 특징에 따라 무작위 추출알고리즘을 이용하여 무작위적으로 설정된다. 먼저 (0,1)에 속한 실수의 난수값이 생성된후 이를 사전 설정된 기준확률값 (여기서는 이상적인 스킵 리스트의 생성을 위해 $p=1/2$ 로 설정)보다 작고 동시에 생성하고자 하는 해당 노드의 레벨이 MaxLevel보다 작을시 해당 노드의 레벨을 증가된 그리하지 않을 경우에는 레벨 0값을 갖는 노드를 생성한다.

<69> 이와 같이, 스킵 리스트에서 프리픽스 범위에 따른 노드를 구성한 후에, 라우트 엔트리를 포함하는 라우팅 테이블이 구축된다. 라우팅 테이블의 구축에 관하여, 라우터가 프리픽스 길이가 30인 라우트를 갖는 경우에 대해 도 10a를 예로 들어 설명하기로 한다.

<70> 도 10a를 참조하면, 스킵 리스트에서 프리픽스 범위가 32~24인 제1노드(220)가 검색되고 이후 추가될 라우트 엔트리가 해시 테이블의 형태로 삽입된다. 스킵 리스트의 노드는 해당 프리픽스 길이에 따라 각기 다른 해시 테이블을 가진다. 동일한 절차에 따라, 도 10b에 도시된 바와 같이 프리픽스 범위가 17~12인 노드(240)하에서 프리픽스 길이가 15인 해시 테이블이 생성됨으로, 프리픽스 길이가 15인 라우트가 추가될 수 있다.

<71> 이러한 라우팅 테이블 생성 절차를 전체적으로 간략히 살펴보면 도 10에 도시된 바와 같은 과정을 가진다. 도 11은 본 발명의 일 실시예에 라우팅 테이블 구축 과정의 개략적인 흐름도이다. 먼저, 11a단계에서 스킵 리스트내의 모든 노드를 다룰 수 있도록 최대 레벨(MaxLevel)을 가진 헤더 노드를 생성한다. 이후 11b단계에서는 고정 분할된 프리픽스 범위를 가진 노드가 상기 리스트에 삽입된다. 상기 11b단계에서는 동작의 간략화를 위해, 본 발명의 일 실시예에서는 상기와 같이 고정 분할된 프리픽스 범위를 제안하였으나, 실제 프리픽스 길이 분산이 반영되도록 적응적으로 분할된 프리픽스 범위를 설정할 수도 있다. 이후 11c단계에서는 프리픽스 범위에 해당하는 노드에서 각각의 주어진

라우트 엔트리를 위한 해시 테이블이 생성된다. 본 발명에서는 종래의 일반적인 트리 또는 트라이 구조와 같은 최소 프리픽스 길이로부터의 검색이 아니라 롱기스트 프리픽스 길이를 가진 라우트로부터 시작하는 검색을 제공한다.

<72> 한편, 라우터는 이웃에 접속된 다른 라우터로부터 수시로 라우트 변경 정보를 수신한다. 이러한 변경은 현재의 라우팅(또는 전송) 테이블에 신속히 반영되어야 함이 바람직하다. 이러한 라우팅 테이블의 갱신 작업에 대하여 이하 도 12를 참조하여 설명하기로 한다. 주어진 변경된 라우트를 라우팅 테이블에 반영하기 위하여, 먼저 12a단계에서 프리픽스 길이가 일치하는 노드를 발견한다. 이후 12b단계에서는 해당 노드에서 변경 라우트와 동일한 프리픽스 길이를 가진 해시 테이블을 검색한다. 해당 해시 테이블이 발견되면, 이를 12c단계에서 확인하여 이후 12d단계로 진행한다. 12d단계에서는 해당 해시 테이블 내의 갱신될 라우트를 변경 또는 삭제한다. 만약 해당 해시 테이블을 존재하지 않을 경우에 이를 상기 12c단계에서 확인하여 이후 12e단계로 진행하며, 12e단계에서는 변경 라우트와 동일한 프리픽스 길이를 가진 해시 테이블을 생성한다. 이후 12f단계에서는 생성된 해시 테이블에 주어진 라우트를 삽입한다. 이러한 동작에 의해 변경된 라우트가 해당 해시 테이블에 갱신(생성, 삭제, 변경)될 수 있다.

<73> 도 13, 14는 각각 본 발명의 일 실시예에 따른 라우트 검색 및 갱신 과정을 설명하기 위한 도면이다. 먼저 도 13을 참조하여 프리픽스 길이가 15인 라우트의 검색 과정을 설명하기로 한다. 먼저, 동일한 프리픽스 길이를 가진 노드를 발견한다. 검색은 헤더 노드(210)의 최상위 레벨(레벨 4)에서부터 시작한다(도 13의 (1)단계). 상기 헤더 노드(210)의 최상위 레벨은 프리픽스 길이 범위가 8~1인 노드(260)를 포인트 한다(도 3의 (2)단계). 이후 찾아간 노드(즉 노드(280))의 프리픽스 길이 범위와 라우트의 프리픽스

길이를 비교하는 동작이 수행된다. 키 값이 일치하지 않을 경우, 헤더 노드(210)의 레벨이 한단계 감소되고, 다음 레벨의 포인터는 프리픽스 길이 범위가 17~12인 노드(240)를 가리키게 된다(도 13의 (3), (4), (5)단계). 결국 해당 노드(240)에서 변경된 라우트의 프리픽스 길이와 일치하는 해시 테이블에 프리픽스가 발견된다(도 13의 (6), (7), (8)단계). 만약 해시 테이블이 존재하지 않으면, 상기 테이블은 새로운 라우트를 추가하도록 생성된다.

<74> 도 14에는 다음 홉(hop) 인터페이스가 '11'인 '111011...1' 프리픽스가 해시 테이블에서 삭제되는 다수의 단계가 개시된다. 각 단계에서 해당 프리픽스를 검색하는 단계는 상기 도 13에 도시된 과정과 유사하게 수행된다.

<75> 이하 상기한 바와 같은 라우팅 테이블에서 특정 목적지 IP 어드레스의 경로를 검색하는 라우트 룩업 동작을 첨부 도면을 참조하여 상세히 설명하기로 한다.

<76> 도 15는 본 발명의 일 실시예에 따른 라우트 룩업 진행 과정을 나타낸다. 먼저 15a 단계에서는 IP 헤더에 캡슐화된(encapsulated) 주어진 목적지 어드레스에 대해, 스킵 리스트의 제1노드가 검색된다. 이후 과정은 목적지 어드레스와 각 해시 테이블이 감소되는 순서로 비교된다. 만약 비교한 해시 테이블이 목적지 어드레스를 포함하지 않으면, 인접한 프리픽스 길이가 하나 감소되는 해시 테이블로 이동하거나 스킵 리스트의 다른 노드로 이동한다.

<77> 이러한 동작을 다시 도 15를 참조하여 보다 상세히 설명하면, 상기 15b단계에서 해시 테이블을 검색하여 15c단계에서 해시 테이블이 존재하면 이를 15c단계에서 확인하여 이후 15d단계로 진행하고 일치하는 해시 테이블의 존재하면 15h단계로 진행한다. 15d단계에서는 주어진 목적지 어드레스와 일치하는 해시 테이블내의 프리픽스를 찾게 되며,

프리픽스가 일치하면 이를 15e단계에서 확인하여 이후 일치한 프리픽스를 롱기스트 프리픽스로서 리턴(return)한다.

<78> 한편, 상기 15e단계에서 프리픽스가 일치하지 않으면 이후 15g단계로 진행하여 목적지 어드레스의 비트 스트링을 1 감소하고 이후 15h단계로 진행한다. 15h단계에서는 현재 프리픽스 길이보다 1 감소된 다음 해시 테이블로 이동하여 다음 해시 테이블이 존재하면 이를 15i단계에서 확인하여 상기 15b단계로 진행함으로 상기의 과정을 반복 진행하게 된다. 한편 상기 15i단계에서 다음 해시 테이블이 존재하지 않을 경우 15j단계로 진행하여 스킵 리스트에서 인접 노드로 이동하고, 상기 15b단계로 진행하여 인접 노드에서 상기의 과정을 반복 진행하게 된다. 이와 같이, 본 발명은 종래의 통상적인 루트 노드로부터 시작하는 검색과는 달리, 자체 해시 테이블에서 롱기스트 프리픽스 범위로부터 검색을 시작하는데 기반을 둔다.

<79> 도 16은 본 발명의 일 실시예에 따른 전체 록업 작업 과정을 설명하기 위한 도면이다. 도 16에서는 프리픽스 길이가 15이며 '1110101...1'인 목적지 어드레스의 록업을 예로 들었다.

<80> 먼저 상기에 언급한 바와 같이 스킵 리스트에서 노드를 찾는다. 주어진 목적지 어드레스에 의해 라우트 록업이 헤더 노드(210)로부터 시작된다(도 16의 (1)단계). 이후 헤더 노드(210)에서 레벨이 0레벨까지 감소된다(도 16의 (2)단계). 이후 헤더 노드(210)의 포인터가 가리키는 제1노드(220)를 거치게 된다(도 16의 (3)단계). 이후 제1노드(220)에서 먼저, 프리픽스 길이가 30 [2]인 해시 테이블(222)이 원래 목적지 어드레스와 비교된다(도 16의 (4)단계). 이러한 단계에서 목적지 어드레스를 찾지 못한 경우, 도 17에 도시된 바와 같은 원래 목적지 어드레스에서 LSB(Least Significant Bit)로부터 1비

트 스트링이 감소된 어드레스와 다음 해시 테이블(프리픽스 길이가 31)을 비교하며, 만약 이 경우에도 목적지 어드레스를 찾지 못한 경우에는 원래 목적지 어드레스 2비트 스트링이 감소된 어드레스와 그 다음 해시 테이블(프리픽스 길이가 30)을 비교하게 된다. 이러한 과정은 일치하는 해시 테이블을 찾지 못한 경우에, 도 16의 (5)단계에서 도시된 바와 같은 제1노드(220)에서 프리픽스 길이가 26인 마지막 해시 테이블(224)을 비교할 때까지 반복 수행된다. 만약 제1노드(220)에서 룩업이 실패하면 이후 스킵 리스트에서 포인터는 다음 제2노드(230)를 가리키게 된다. 이때 상기 제2노드(230)에는 어떠한 해시 테이블도 존재하지 않을 수 있다. 그럴 경우에는 상기 스킵 리스트에서 다음 제3노드(240)로 이동하게 된다(도 16의 (6)단계). 이후 제3노드(240)에서도 상기 제1노드(220)에서와 같이 해시 테이블을 비교하는 작업이 이루어지며, 결국 도 16의 (7)단계에서 11비트 스트링이 제거된 목적지 어드레스와 프리픽스 길이가 15인 해시 테이블의 비교가 이루어진다. 이러한 비교에 의해 일치가 발생하면, 프리픽스 길이 15인 해당 해시 테이블에서 롱기스트 프리픽스가 발견된다(도 16의 (8)단계). 이에 따라 라우트 룩업 동작이 성공적으로 종료된다.

<81> 이하 상기한 스킵 리스트의 구축과, 라우트 엔트리의 반영 및 라우트 룩업에 관한 프로그램 소스를 개시한다.

<82> 먼저 하기 프로그램 1, 2, 3은 스킵 리스트의 구축에 관한 것이다.

<83> [프로그램 1 : Buildup operation]

<84>

```

var px: bitstring init b'0'; (* prefix *)
interface: integer init 0; (* next hop interface *)
W: integer init 32; (* # of address bits, IPv4 *)
k: integer init 8; (* fixed prefix range, e.g., 8 *)
hrange: integer init W; (* upper bound of prefix range *)
lrange: integer init W - k + 1; (* lower bound of prefix range *)
MaxLevel: integer init  $\lceil \lg(W/k) \rceil$ ; (* assuming  $p = 1/2$  *)

Buildup(list) (* constructing a skip list *)
begin
  L :=  $\lceil W/k \rceil$ ;
  while L ≥ 0 do
    Insert_Node(list, hrange, lrange);
    L := L - 1;
    hrange := lrange - 1;
    lrange := lrange - k + 1;
  end while
end

```

<85>

[프로그래밍 2 : Inserting a node in a skip list]

<86>

```

Insert_Node(list, hrange, lrange) (* inserting a node *)
begin
  update[0..MaxLevel];
  x := list → header;
  for i := list → level downto 0 do
    while [hrange, lrange]  $\not\in$  x → forward[i] → key do
      x := x → forward[i];
    end while
    update[i] := x;
  end for
  x := x → forward[0];
  if [hrange, lrange]  $\in$  x → key then
    x → value := [hrange, lrange];
  else
    v := RandomLevel();
    if v > list → level then
      for i := list → level + 1 to v do
        update[i] := list → header;
      end for
      list → level := v;
    end if
    x := MakeNode(v, [hrange, lrange]);
    for i := 0 to level do
      x → forward[i] := update[i] → forward[i];
      update[i] → forward[i] := x;
    end for
  end if
end

```

<87>

[프로그래밍 3 : Randomizing level]

```

<88>      RandomLevel() (* randomly deciding a level of a node *)
      begin
        v := 0;
        r := Random (); (* r ∈ [0, 1) *)
        while r < p and v < MaxLevel do
          v := v+1;
        end while
        return v;
      end

```

<89> 하기 프로그램 4, 5, 6, 7은 스킵 리스트에서 라우트 엔트리의 반영에 관한 것으로
 , 프로그램 4는 검색 작업, 프로그램 5는 삽입 작업, 프로그램 6은 삭제 작업, 프로그램
 7은 갱신 작업에 관한 것이다.

<90> [프로그램 4 : Search operation]

```

<91>      Search(list, px) (* searching for a node with prefix px in a skip list *)
      begin
        x := list → header;
        for i := list → level downto 0 do
          while px ∉ x → forward[i] → key do
            x := x → forward[i];
          end while
        end for
        x := x → forward[0];
        if x → key = px then
          for j := hrange downto lrange-1 do
            if hash-get (j, px) then
              return hash-get (j, px);
            end if
          end for
          return error;
        end if
      end

```

<92> [프로그램 5: Insert operation]

<93>

```
Insert(list, prefix, interface) (* inserting a route entry *)
begin
  if search (list, prefix) then
    hash-insert (prefix, interface);
  else
    return error;
  end if
end
```

<94>

[프로그램 6 : Delete operation]

<95>

```
Delete(list, prefix, interface) (* deleting a route entry *)
begin
  if search (list, prefix) then
    hash-delete (prefix, interface);
  else
    return error;
  end if
end
```

<96>

[프로그램 7 : Update a route entry]

<97>

```
Update(list, prefix, interface) (* updating a route entry *)
begin
  if search (list, prefix) then
    hash-update (prefix, interface);
  else
    return error;
  end if
end
```

<98>

하기 프로그램 8은 라우트 룩업에 관한 것이다.

<99>

[프로그램 8 : Lookup operation]

```

<100>      Lookup(list, destaddr) (* looking for a route with the longest prefix *)
      begin
         $x := list \rightarrow header;$ 
         $x := x \rightarrow forward[0];$ 
        for  $i := \lceil W/k \rceil$  downto 0 do
          for  $j := hrange$  downto  $lrange-1$  do
            if compare( $destaddr, hash-get(j, destaddr)$ ) then
              return  $hash-get(j, destaddr);$ 
            end if
          end for
           $x := x \rightarrow forward[i];$ 
        end for
        return error;
      end

```

<101> 이하 첨부 도면을 참조하여, 본 발명에 따른 라우팅 방식의 효율을 설명하기로 한다. 도 7a, 7b는 인터넷 상에서의 프리픽스 길이 분포에 대한 실측치를 나타낸 그래프이다. 도 7a 및 7b의 그래프에 나타난 데이터는 인터넷상에 네트워크 및 네트워킹 프로토콜의 성능을 연구하는 IPMA(Internet Performance Measurement and Analysis) 프로젝트에서 최근 MAE-EAST와 PAIX NAP(Network Access Point)에서 수집한 자료를 프리픽스 길이별 라우트 검색시 발생빈도를 그래프로 나타낸 것이다.

<102> IPMA 프로젝트에서 수집한 통계 데이터는 특히 라우트 록업 알고리즘 개발자의 연구결과를 상호 비교 분석하는 원천자료로 널리 활용된다. MAE-EAST NAP에 있는 라우터는 현재 가장 많은 라우트 엔트리를 포함하며 주로 대규모 기간(backbone) 라우터의 모델로서 사용된다. 그리고 PAIX NAP은 현재 IPMA 프로젝트에서 수집된 최소의 라우트 엔트리를 포함하고 있으며 주로 엔터프라이즈(enterprise) 라우터의 모델로 이용된다.

<103> 도 7a 및 7b의 그래프에 도시된 바와 같이 프리픽스 길이가 16~28일 때 가장 많이 참조가 됨을 나타내는 국소성 패턴(locality pattern)을 보여 준다. 따라서 이러한 관측 결과는 본 발명에서 제시된 변형 스킵 리스트에서 포함될 노드의 수는 상당히 감소할 것

을 보여준다. 참조문헌 11에서 인용한 바와 같이 IPMA(24번 문헌)에서의 또 다른 흥미로운 결과는 하루에 BGP에 의해 추가되거나 또는 철회된 라우팅 프리픽스의 수는 1,899,851(1997년 11월 1일 현재)에 달한다. 이는 25 prefix-updates/sec를 요구해야 할 만큼 긴박한 것이며 변경된 라우팅 프리픽스를 라우팅 테이블 또는 포워딩 테이블에 신속히 반영하는 것은 검색시간외에 중요 요소임을 확인할 수 있다.

<104> 이하 본 발명에서 따른 알고리즘과 종래의 알고리즘과의 차이점을 하기 표 2를 통해 검토해 보기로 한다. 통상 알고리즘에 대한 비교 분석은 해당 알고리즘을 처리하는데 요구되는 시간 또는 사용하는 메모리에 대한 'worst-case complexity (Big-O notation 사용)'를 상호 비교한다. 먼저, N개의 라우트 엔트리가 존재하고, 각 어드레스를 표현하기 위해서는 W개의 어드레스 비트가 요구되며, k는 알고리즘 의존성 상수(constant)를 의미한다고 할 때, 하기 표 2는 본 발명에 따른 알고리즘과의 요구되는 복잡도 (complexity)를 나타내었다.

<105> 【표 2】

	<i>Build</i>	<i>Insert</i>	<i>Delete</i>	<i>LPM Search</i>	<i>Memory</i>
Array	$O(N)$	$O(N^2)$	$O(N^2)$	$O(N)$	$O(N)$
Hash table	$O(N)$	$O(1)$	$O(1)$	$O(N)$	$O(N)$
Binary search tree	$O(N \lg N)$	$O(N)$	$O(N)$	$O(\lg(2N))^a$	$O(N)$
Trie	$O(NW)$	—	—	$O(W)$	$O(NW)$
Radix trie	$O(NW)$	—	—	$O(W)$	$O(N)$
PATRICIA trie	$O(NW)$	—	—	$O(W^2)$	$O(N)$
DP trie	$O(NW)$	$O(W)$	$O(W)$	$O(W)$	$O(N)$
LC trie	$O(NW)$	—	—	$O(W)$	$O(N)$
Hashed radix tree	$O(NW)$	—	—	$O(W/k)$	$O(NW)$
Basic BST scheme without backtracking	$O(N \lg W)$	$O(\lg N)$	$O(\lg N)$	$O(\lg W)$	$O(N \lg W)$
Asymmetric BST	$O(N \lg W)$	$O(N)$	$O(N)$	$O(\lg W)$	$O(N \lg W)$
Rope search	$O(NW)$	$O(N)$	$O(N)$	$O(\lg W)$	$O(N \lg W)$
Ternary CAMs	$O(N)$	$O(1)$	$O(1)$	$O(1)$	$O(N)$
Complete prefix tree	$O(N \lg W)$	—	—	$O(W/k)$	$O(N)$
Binary hash table search	$O(N \lg W)$	—	—	$O(\lg(W))$	$O(N)$
Multway and multicolumn search	$O(NW)$	$O(N)$	$O(N)$	$O(\log_k N)$	$O(N)$
Large memory architecture	$O(N)$	—	—	$O(W/k)$	$O(N)$
Skip list	$O(N \lg N)$	$O(\lg N)$	$O(\lg N)$	$O(N \lg N)$	$O(N)$
Ours	$O(N \lg(W/k))$	$O(\lg(W/k))$	$O(\lg(W/k))$	$O(W)$	$O(N)$

<106> 상기 표 2에서 상용대수(logarithm with base = 10)와 자연대수(natural logarithm = logarithm with base e)와의 혼동을 피하기 위해서, \log_2 (base가 2인 대수)는 lg로 표현한다.

<107> IPv4에서는 어드레스 비트 수 W는 32에 해당되고 IPv6에서는 128에 해당한다.
MAP-EAST 또는 PAIX에서는 대략적으로 라우트 엔트리 수 N은 각각 50,000과 6,000에 해당된다. 상수 k는, 본 발명에서 제시된 스킵 리스트의 경우에, 프리픽스 길이 범위를 8이라 하였을 때 IPv4에서는 4가 되며 IPv6에서는 16이 된다.

<108> 상기한 바와 같이, 본 발명은 기존의 이진 검색 트리(binary search tree) 및 트라이와 이의 유사한 변형(예를 들면, 'Patricia' 또는 'LC-tries')에 비해서 테이블의 구축, 검색, 추가 및 삭제를 비교할 때 월등함을 알 수 있다. 아울러, 현재 비교적 여타의 알고리즘에 비해 검색시간이 향상된 'Binary hash table search(14번 문헌)' 및 'Complete prefix trie(13번 문헌)'에 비해 검색시간 및 추가 또는 삭제에 있어서도 월등함을 알 수 있다. 그 외에 순수한 스킵 리스트(pure skip list)에 대한 복잡도 비교시에도 상당히 우수함을 상기 표 2로부터 확인할 수 있다.

<109> 상기한 바와 같이, 본 발명은 본 발명은 무작위 추출 알고리즘인 스킵 리스트와 해서 테이블 등을 이용하여 인터넷프로토콜 어드레스의 프리픽스 범위에 따라 노드가 생성되며, 각 노드에서 각각 설정된 프리픽스 길이에 따른 해시 테이블 형태로 라우팅 엔트리가 저장되는 스킵 리스트를 이용한 라우팅 방식을 제공하여, 최소한의 경로검색시간 제공 및 관련 테이블의 효율적인 관리를 통해 최단 시간내 경로검색이 가능하도록 한다. 이러한 본 발명은 향후 모든 라우터에서 탑재 운영될 IPv6의 경우에도 동일하게 적용될 수 있다.

<110> 한편 상기한 본 발명의 상세한 설명에서는 구체적인 실시예에 관해 설명하였으나, 이는 본 발명을 제한하려는 의도가 아니라 단지 설명을 위한 것이며, 다양한 변형이 본 발명의 범위를 벗어나지 않고 실시될 수 있음은 당업자에게 명백할 것이다. 따라서 본 발명의 범위는 상술한 실시예에 의하여 한정되는 것이 아니라 청구범위와 청구범위의 균등한 것에 의하여 정하여져야 할 것이다.

【발명의 효과】

<111> 상기한 바와 같이, 본 발명은 기존에 제시된 여타 알고리즘들에 비해 라우팅 테이블 또는 포워딩 테이블의 구축, 추가, 삭제 및 검색에서 특정 동작 향상에만 치우치지 않고 전체적으로 고르게 뛰어난 복잡도 분석 결과로부터 확인할 수 있다. 더욱이, 분산 또는 병렬형태의 라우터 구조로 가는 현 추세에서 포워딩 엔진에 위치한 포워딩 테이블의 재구성(re-build up)이 아닌 변경된 경로에 한해서만 추가 및 삭제할 경우에는 라우팅 프로세서와 포워딩 엔진간의 메모리 대역폭 감소 등에 따른 간접적인 기대 효과를 얻을 수 있다. 아울러, 경로검색시간 단축 및 라우팅 테이블에 대한 신속한 관리는 라우터 자체의 성능향상 및 이외 향후 QoS, 멀티캐스팅, IPSec 등의 구현 등에 간접적인 성능향상 기여에 대한 부가효과를 기대할 수 있다.

<112> [참조 문헌 리스트]

<113> 1. Keshav, S. and Sharma, R., Issues and Trends in Router Design, *IEEE Communications Magazine*, pages 144-151, May, 1998.

<114> 2. Kumar, V. and Lakshman, T. and Stiliadis, D., Beyond Best Effort: Router

Architectures for the Differentiated Services of Tomorrow's Internet, , pages 152-164, May, 1998.

- <115> 3. Chan, H., Alnuweiri, H. and Leung, V., A Framework for Optimizing the Cost and Performance of Next-Generation IP Routers, *IEEE Journal of Selected Areas in Communications*, Vol. 17, No.6, pages 1013-1029, June 1999.
- <116> 4. Partridge, C. et al., A 50-Gb/s IP Router, *IEEE/ACM Trans. on Networking*, vol. 6, no.3, pages 237-248, 1998.
- <117> 5. Metz, C., IP Routers: New Tool for Gigabit Networking, *IEEE Internet Computing*, pages 14-18, Nov.-Dec., 1998.
- <118> 6. Asthana, A., Delph, C., Jagadish, H., and Krzyzanowski, P., 'Towards a gigabit IP router', *J. High Speed Network*, vol. 1, no.4, pages 281-288, , 1992.
- <119> 7. RFC 1518, An Architecture for IP Address Allocation with CIDR, Sept. 1993
- <120> 8. RFC 1517, Applicability Statement for the Implementation of Classless Inter-Domain Routing (CIDR), Sept. 1993
- <121> 9. Doeringer, W., Karjoth, G. and Nassehi, M., Routing on Longest-Matching Prefixes, *IEEE/ACM Trans. on Networking*, vol.4, no.1, pages 86-97, Feb., 1996.
- <122> 10. Degermark, M., Brodnik, A., Carlsson, S. and Pink, S., Small Forwarding Tables for Fast Routing Lookups, In *Proceedings of ACM SIGCOMM '97*, pages 3-14, Cannes, France, 1997.
- <123> 11. Srinivasan, V. and Varghese, G., Faster IP Lookups using Controlled

Prefix Expansion, In *Proceedings of ACM Sigmetrics '98 Conf.*, pages 1-11, 1998.

- <124> 12. Lampson, B., Srinivasan, V. and Varghese, G., IP Lookups using Multiway and Multicolumn Search, In *IEEE Infocom*, pages 1248-1256, 1998.
- <125> 13. Tzeng, H. and Przygienda, T., On Fast Address-Lookup Algorithms, *IEEE Journal on Selected Areas in Communications*, Vol. 17, No. 6, pages 1067-1082, June, 1999.
- <126> 14. Waldvogel, M., Varghese, G., Turner, J. and Plattner, B., Scalable High Speed IP Routing Lookups, In *Proceedings of ACM SIGCOMM '97*, Cannes, France, pages 25-37, 1997.
- <127> 15. Waldvogel, M., Varghese, G., Turner, J. and Plattner, B., Scalable Best Matching Prefix Lookups, In *Proceedings of PODC '98*, Puerto Vallarta, page, 1998.
- <128> 16. Kijkanjanarat, T. and Chao, H., Fast IP Lookups Using a Two-trie Data Structure, In *Proceedings of Globecom'99*, 1999.
- <129> 17. Nillson, S. and Karlsson, G., IP-Addresses Lookup Using LC-Tries, *IEEE Journal on Selected Areas in Communications*, Vol.17, No. 16, pages 1083-1092, 1999.
- <130> 18. Crescenzi, P., Dardini, L. and Grossi, R., 'IP Address Lookup Made Fast and Simple', Technical Report TR-99-01, Dipartimento di Informatica, University a Di Pisa, 1999.
- <131> 19. Gupta, P., Lin, S. and McKeown, N., Routing Lookups in Hardware at Memory

Access Speeds, In *Proceedings of IEEE INFOCOM '98 Conf.*, pages 1240-1247,, 1998.

<132> 20. McAuley, A. and Francis, P., Fast Routing Table Lookup Using CAMs, In *Proceedings of IEEE INFOCOM '93*, Vol.3, pages 1382-1391, 1993.

<133> 21. Huang, N. and Zhao, S., A Novel IP-Routing Lookup Scheme and Hardware Architecture for Multigigabit Switching Routers, *IEEE Journal on Selected Areas in Communications*, Vol. 17, No. 6, pages 1093-1104, June, 1999.

<134> 22. Pugh, W., Skip Lists: A Probabilistic Alternatives to Balanced Trees, *CACM* 33(6), pages 6689-676, 1990.

<135> 23. Sleator, D. and Tarjan, R., Self-Adjusting Binary Search Trees, *JACM*, Vol. 32, No. 3, July 1985.

<136> 24. IPMA (Internet Performance Measurement and Analysis), <http://nic.merit.edu/ipma>

<137> 25. Cormen, T., Leiserson, C. and Rivest, R., *Introduction to Algorithms*, McGraw-Hill, New York, 1990.

【특허청구범위】**【청구항 1】**

스킵 리스트를 이용하여 인터넷프로토콜 어드레스 검색을 위한 라우팅/포워딩 테이블 구축 방법에 있어서,

상기 인터넷프로토콜 어드레스의 프리픽스 길이 범위를 미리 설정된 방식으로 분할하는 과정과,

상기 프리픽스 길이 범위의 분할된 클러스터(cluster) 갯수에 따른 최대 레벨을 가지며, 상기 스킵 리스트내의 다른 모든 노드를 포인팅하기 위한 헤더 노드를 생성하는 과정과,

상기 분류된 프리픽스 길이 범위를 키로서 각각 가지며 상기 분할된 클러스터 갯수에 해당하는 수의 서브 노드를 생성하는 과정을 가짐을 특징으로 하는 라우팅/포워딩 테이블 구축 방법.

【청구항 2】

제1항에 있어서, 상기 각각의 서브 노드에서 해당 프리픽스 길이 범위에 따른 각각의 프리픽스 길이별 해시 테이블을 구비하여, 상기 해시 테이블에 상기 각각의 프리픽스 길이에 해당하는 라우트 엔트리를 저장하는 과정을 더 가짐을 특징으로 하는 라우팅/포워딩 테이블 구축 방법.

【청구항 3】

제2항에 있어서, 상기 라우트 엔트리는 32비트 또는 128비트임을 특징으로 하는 라우팅/포워딩 테이블 구축 방법.

【청구항 4】

제1항 또는 제2항에 있어서, 상기 서브 노드의 상기 헤더 노드에서의 레벨은 무작위적으로 설정함을 특징으로 하는 라우팅/포워딩 테이블 구축 방법.

【청구항 5】

제1항 또는 제2항에 있어서, 상기 프리픽스 길이 범위의 분할은 고정 또는 가변적으로 설정함을 특징으로 하는 라우팅/포워딩 테이블 구축 방법.

【청구항 6】

제1항 또는 제2항에 있어서, 상기 프리픽스 길이 범위의 분할은 상기 다수의 서브 노드를 통해 모든 프리픽스 길이 범위가 커버되도록 분할함을 특징으로 하는 라우팅/포워딩 테이블 구축 방법.

【청구항 7】

제1항 또는 제2항에 있어서, 상기 라우팅/포워딩 테이블은 라우팅 프로세서 및 포워딩 엔진에 구비되는 라우팅/포워딩 테이블임을 특징으로 하는 라우팅/포워딩 테이블 구축 방법.

【청구항 8】

스킵 리스트를 이용한 인터넷프로토콜 라우팅/포워딩 테이블 생성 방법에 있어서, 상기 스킵 리스트내의 모든 노드를 다루기 위해 각각의 노드를 포인팅하는 헤더 노드를 생성하는 과정과,

미리 설정된 방식으로 상기 인터넷프로토콜 어드레스의 프리픽스 범위의 분할에 따라 각각 분할된 상기 프리픽스 범위를 키로 하는 다수의 서브 노드를 생성하는 과정과,

상기 각각의 서브 노드에서 해당 프리픽스 길이 범위에 따른 각각의 프리픽스 길이별 라우트 엔트리를 저장하기 위하여, 각각의 프리픽스 길이별 해시 테이블을 생성하는 과정을 가짐을 특징으로 하는 라우팅/포워딩 테이블 생성 방법.

【청구항 9】

제8항에 있어서, 상기 해시 테이블에 해당 프리픽스에 일치하는 라우트 엔트리를 저장하는 과정을 더 가짐을 특징으로 하는 라우팅/포워딩 테이블 생성 방법.

【청구항 10】

제8항 또는 제9항에 있어서, 상기 서브 노드의 상기 헤더 노드에서의 레벨은 무작위적으로 설정함을 특징으로 하는 라우팅/포워딩 테이블 생성 방법.

【청구항 11】

제8항 또는 제9항에 있어서, 상기 프리픽스 길이 범위의 분할은 고정적 또는 가변적으로 설정함으로 특징으로 하는 라우팅/포워딩 테이블 생성 방법.

【청구항 12】

제8항 또는 제9항에 있어서, 상기 프리픽스 길이 범위의 분할은 상기 다수의 서브 노드를 통해 모든 프리픽스 길이 범위가 커버되도록 분할함을 특징으로 하는 라우팅/포워딩 테이블 생성 방법.

【청구항 13】

제8항 또는 제9항에 있어서, 상기 라우트 엔트리는 32비트 또는 128비트임을 특징으로 하는 라우팅/포워딩 테이블 생성 방법.

【청구항 14】

제8항 또는 제9항에 있어서 상기 헤더 노드는 $+\infty$ 키 값을 가지며 최대 레벨개의 색인되는 전송 포인터를 가짐을 특징으로 하는 라우팅/포워딩 테이블 생성 방법.

【청구항 15】

제8항 또는 제9항에 있어서 상기 라우팅/포워딩 테이블은 라우팅 프로세서 및 포워딩 엔진에 구비되는 라우팅/포워딩 테이블임을 특징으로 하는 라우팅/포워딩 테이블 생성 방법.

【청구항 16】

인터넷프로토콜 어드레스의 프리픽스 범위 할당에 따라 노드가 생성되며, 각 노드에서 각각 설정된 프리픽스 길이에 따른 해시 테이블의 형태로 라우팅 엔트리가 저장되는 스킵 리스트를 이용한 라우팅/포워딩 테이블의 검색 방법에 있어서,

상기 노드 중 검색할 라우트의 프리픽스 길이에 해당하는 프리픽스 범위가 설정된 노드를 찾는 과정과,

상기 찾은 노드에서 상기 검색할 라우트와 동일한 프리픽스 길이를 가진 상기 해시 테이블을 찾는 과정과,

상기 찾은 해시 테이블에서 상기 검색 라우트를 발견하는 과정을 가짐을 특징으로 하는 라우팅/포워딩 테이블 검색 방법.

【청구항 17】

제16항에 있어서, 상기 검색할 라우트의 프리픽스 길이에 해당하는 프리픽스 범위가 설정된 노드를 찾는 과정은, 상기 스킵 리스트의 헤더 노드의 최상위 레벨에서부터

포인트하는 노드를 찾아가는 단계와, 상기 찾아간 노드의 프리픽스 길이 범위와 검색 라우트의 프리픽스 길이를 비교하는 단계와, 상기 찾아간 노드의 프리픽스 길이 범위에 상기 검색 라우트의 프리픽스 길이가 해당하지 경우에 상기 헤더 노드의 다음 레벨에서 포인트하는 노드를 찾아가는 단계를 포함하여 이루어짐을 특징으로 하는 라우팅/포워딩 테이블 검색 방법.

【청구항 18】

제16항에 있어서, 상기 스킵 리스트는 헤더 노드와, 감소되는 순서로 미리 설정되는 범위의 상기 키를 각각 가지며, 각 프리픽스 길이에 해당하는 각각의 해시 테이블이 구비되어 상기 해시 테이블에 상기 각각의 프리픽스 길이에 해당하는 라우트 엔트리가 저장되는 다수의 노드를 포함함을 특징으로 하는 라우팅/포워딩 테이블 검색 방법.

【청구항 19】

제16항에 있어서 상기 라우팅/포워딩 테이블은 라우팅 프로세서 및 포워딩 엔진에 구비되는 라우팅/포워딩 테이블임을 특징으로 하는 라우팅/포워딩 테이블 검색 방법.

【청구항 20】

제16항에 있어서, 상기 라우트 엔트리는 32비트 또는 128비트임을 특징으로 하는 라우팅/포워딩 테이블 검색 방법.

【청구항 21】

인터넷프로토콜 어드레스의 프리픽스 범위 할당에 따라 노드가 생성되며, 각 노드에서 각각 설정된 프리픽스 길이에 따른 해시 테이블 형태로 라우팅 엔트리가 저장되는 스킵 리스트를 이용한 라우팅/포워딩 테이블의 갱신 방법에 있어서,

갱신할 라우트의 프리픽스 길이에 해당하는 프리픽스 범위가 설정된 노드를 찾는 과정과,

상기 찾은 노드에서 상기 갱신할 라우트와 동일한 프리픽스 길이를 가진 상기 해시 테이블을 검색하는 과정과,

상기 해당 해시 테이블의 발견시에 해당 해시 테이블에서 해당 라우트를 갱신하는 과정을 가짐을 특징으로 하는 라우팅/포워딩 테이블 갱신 방법.

【청구항 22】

제21항에 있어서, 상기 갱신할 라우트의 프리픽스 길이에 해당하는 프리픽스 범위가 설정된 노드를 찾는 과정은, 상기 스킵 리스트의 헤더 노드의 최상위 레벨에서부터 포인트하는 노드를 찾아가는 단계와, 상기 찾아간 노드의 프리픽스 길이 범위와 상기 갱신할 라우트의 프리픽스 길이를 비교하는 단계와, 상기 찾아간 노드의 프리픽스 길이 범위에 상기 갱신할 라우트의 프리픽스 길이가 해당하지 경우에 상기 헤더 노드의 다음 레벨에서 포인트하는 노드를 찾아가는 단계를 포함하여 이루어짐을 특징으로 하는 라우팅/포워딩 테이블 갱신 방법.

【청구항 23】

제21항에 있어서, 상기 해시 테이블에서 해당 라우트를 갱신하는 과정은 갱신할 라우트를 추가, 변경 또는 삭제하는 과정임을 특징으로 하는 라우팅/포워딩 테이블 갱신 방법.

【청구항 24】

제21항에 있어서,

상기 해당 해시 테이블이 발견되지 않을 시에 상기 갱신할 라우트와 동일한 프리픽스를 가진 해시 테이블을 생성하는 과정과,

상기 생성된 해시 테이블에 상기 갱신할 라우트를 삽입하는 과정을 더 가짐을 특징으로 하는 라우팅/포워딩 테이블 갱신 방법.

【청구항 25】

제21항에 있어서, 상기 스킵 리스트는 헤더 노드와, 감소되는 순서로 미리 설정되는 범위의 상기 키를 각각 가지며, 각 프리픽스 길이에 해당하는 각각의 해시 테이블이 구비되어 상기 해시 테이블에 상기 각각의 프리픽스 길이에 해당하는 라우트 엔트리가 저장되는 다수의 노드를 포함함을 특징으로 하는 라우팅/포워딩 테이블 갱신 방법.

【청구항 26】

제21항 또는 제24항에 있어서 상기 라우팅/포워딩 테이블은 라우팅 프로세서 및 포워딩 엔진에 구비되는 라우팅/포워딩 테이블임을 특징으로 하는 라우팅/포워딩 테이블 갱신 방법.

【청구항 27】

제21항 또는 제24항에 있어서, 상기 라우트 엔트리는 32비트 또는 128비트임을 특징으로 하는 라우팅/포워딩 테이블 갱신 방법.

【청구항 28】

인터넷프로토콜 어드레스의 프리픽스 범위 할당에 따라 노드가 생성되며, 각 노드에서 각각 설정된 프리픽스 길이에 따른 해시 테이블 형태로 라우팅 엔트리가 저장되는 스킵 리스트를 이용한 라우팅/포워딩 테이블의 라우트 록업에 있어서,

상기 스킵 리스트의 제1노드에서부터 시작하여 인접한 노드로 찾아가는 과정과,
해당 노드에서 목적지 어드레스와 각각의 해시 테이블을 비교하는 과정과,
상기 비교 결과 상기 비교하는 해시 테이블이 상기 목적지 어드레스를 포함할 경우
에 상기 일치하는 프리픽스를 롱기스트 프리픽스로 간주하는 과정을 가짐을 특징으로 하
는 라우트 록업 방법.

【청구항 29】

제28항에 있어서, 상기 비교 결과 상기 비교하는 해시 테이블이 상기 목적지 어드
레스를 포함하지 않을 경우에 상기 스킵 리스트에서 상기 해당 노드의 다음 노드로 찾아
가는 과정을 더 가짐을 특징으로 하는 라우트 록업 방법.

【청구항 30】

제28항 또는 제29항에 있어서, 상기 스킵 리스트는 헤더 노드와, 감소되는 순서로
미리 설정되는 범위의 상기 키를 각각 가지며, 각 프리픽스 길이에 해당하는 각각의 해
시 테이블이 구비되어 상기 해시 테이블에 상기 각각의 프리픽스 길이에 해당하는 라우
트 엔트리가 저장되는 다수의 노드를 포함함을 특징으로 하는 라우트 록업 방법.

【청구항 31】

제28항 또는 제29항에 있어서, 상기 해당 노드에서 목적지 어드레스와 각각의 해시
테이블을 비교하는 과정은 상기 목적지 어드레스와 상기 해당 노드의 각각의 해시 테이
블 중 프리픽스 길이가 가장 긴 해시 테이블을 비교하는 단계와, 상기 비교 결과 상기
해시 테이블에서 목적지 어드레스를 찾지 못한 경우 상기 목적지 어드레스에서
LSB(Least Significant Bit)로부터 1비트 스트링씩 감소한 어드레스와 상기 프리픽스 길

이 순서에 따른 각각 다음번재의 해시 테이블을 비교하는 단계를 포함하여 이루어짐을 특징으로 하는 라우트 록업 방법.

【청구항 32】

제28항 또는 제29항에 있어서 상기 라우팅/포워딩 테이블은 라우팅 프로세서 및 포워딩 엔진에 구비되는 라우팅/포워딩 테이블임을 특징으로 하는 라우트 록업 방법.

【청구항 33】

제28항 또는 제29항에 있어서, 상기 라우트 엔트리는 32비트 또는 128비트임을 특징으로 하는 라우트 록업 방법.

【청구항 34】

고속 인터넷 프로토콜 라우터에 있어서,
패킷이 입력 또는 출력되는 포워딩 엔진이 장착된 다수의 라인 카드 모듈과,
상기 패킷을 내부 포트간 스위칭하는 스위치 패브릭과,
헤더 노드와, 인터넷프로토콜 어드레스의 프리픽스 범위 할당에 따라 각각 생성되며 상기 프리픽스 길이 범위를 키로 하고 각각 설정된 프리픽스 길이에 따른 해시 테이블 형태로 라우팅 엔트리가 저장되는 다수의 노드로 이루어지는 스킵 리스트 구조를 갖는 라우팅 테이블을 구비하며, 라우팅 동작을 총괄적으로 제어하는 라우팅 프로세서를 포함하여 구성함을 특징으로 하는 라우터.

【청구항 35】

고속 인터넷 프로토콜 라우터에 있어서,

패킷이 입력 또는 출력되며, 헤더 노드와, 인터넷프로토콜 어드레스의 프리픽스 범위 할당에 따라 각각 생성되며 상기 프리픽스 길이 범위를 키로 하고 각각 설정된 프리픽스 길이에 따른 해시 테이블 형태로 라우팅 엔트리가 저장되는 다수의 노드로 이루어지는 스킵 리스트 구조를 갖는 포워딩 테이블을 구비한 포워딩 엔진이 장착된 다수의 라인 카드 모듈과,

상기 패킷을 내부 포트간 스위칭하는 스위치 패브릭과,

라우팅 동작을 총괄적으로 제어하는 라우팅 프로세서

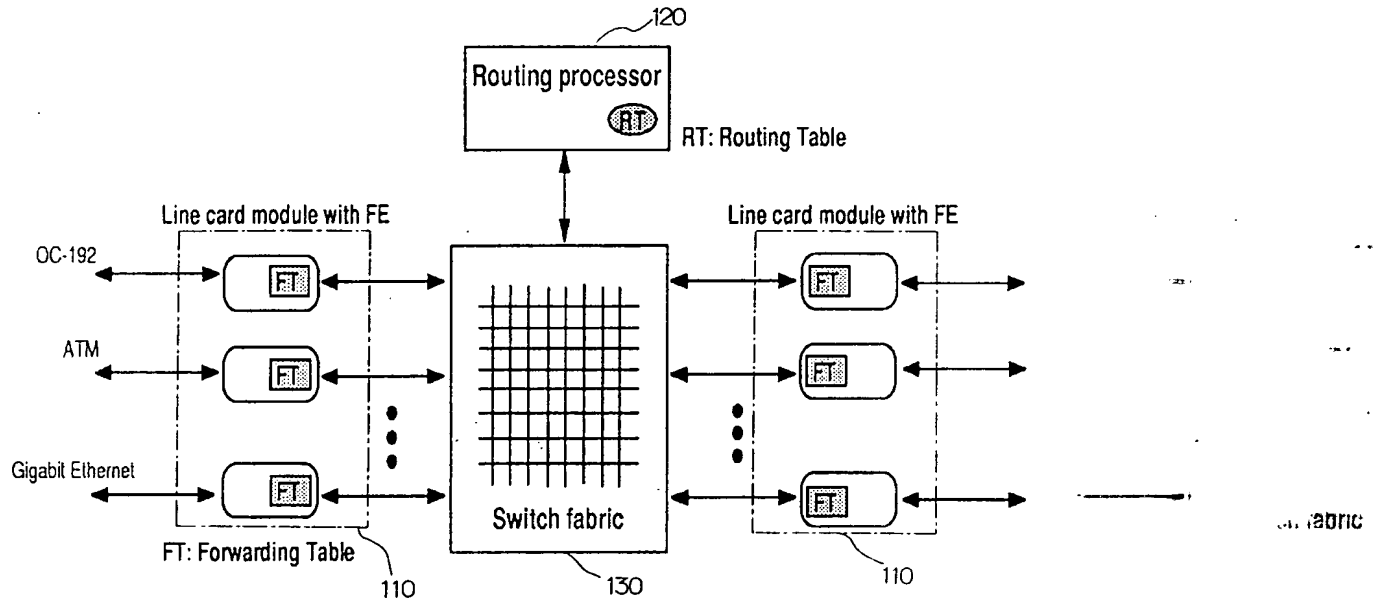
를 포함하여 구성함을 특징으로 하는 라우터.

【청구항 36】

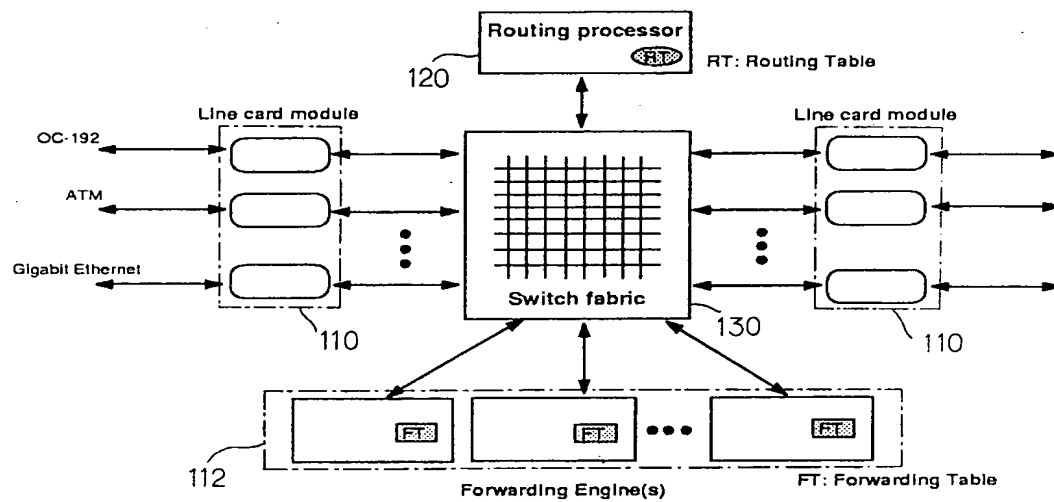
제35항에 있어서, 상기 라우팅 프로세서는 패킷이 입력 또는 출력되며, 헤더 노드와, 인터넷프로토콜 어드레스의 프리픽스 범위 할당에 따라 각각 생성되며 상기 프리픽스 길이 범위를 키로 하고 각각 설정된 프리픽스 길이에 따른 해시 테이블 형태로 라우팅 엔트리가 저장되는 다수의 노드로 이루어지는 스킵 리스트 구조를 갖는 라우팅 테이블을 구비함을 특징으로 하는 라우터.

【도면】

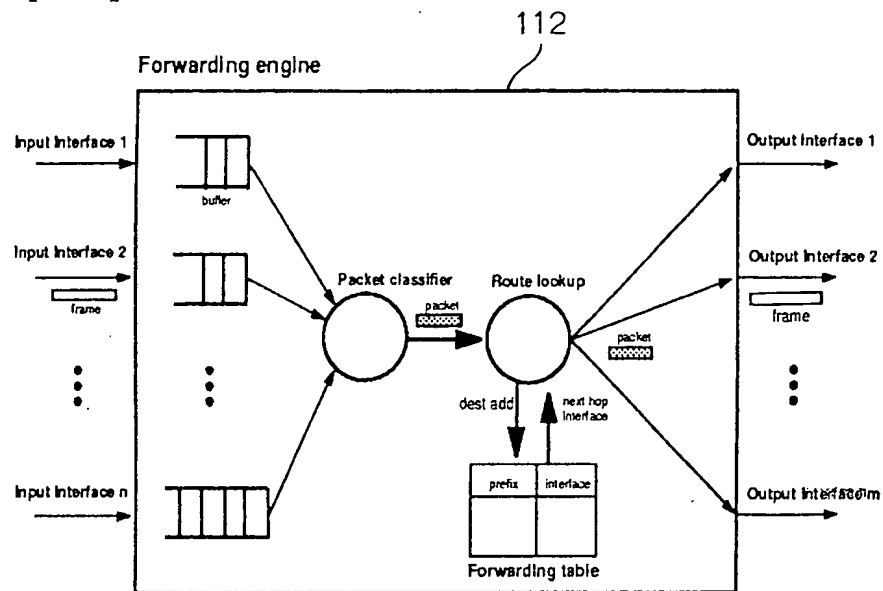
【도 1】



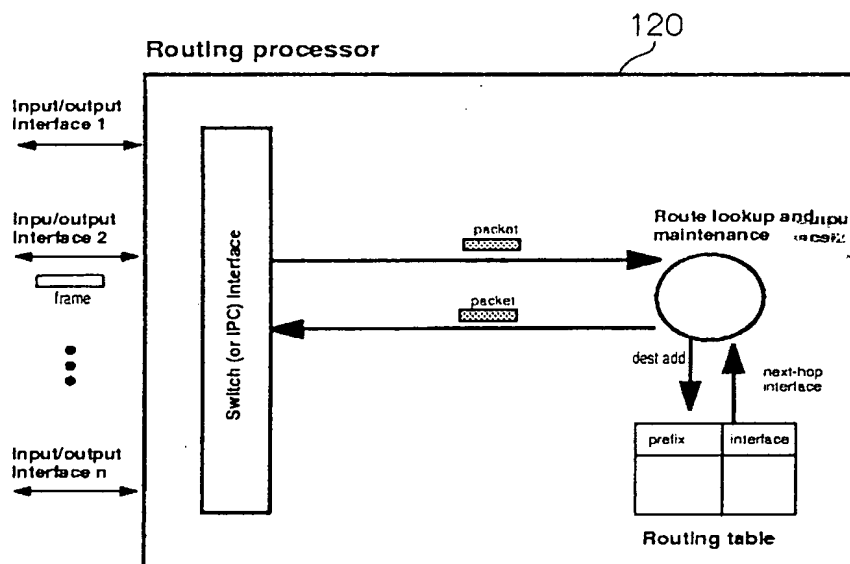
【도 2】



【도 3】



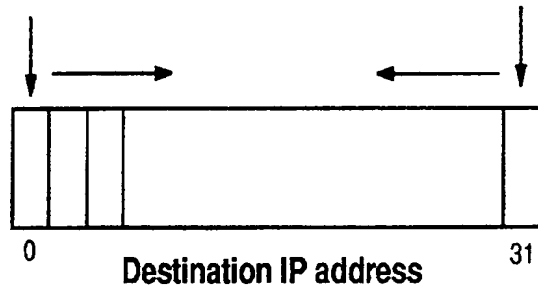
【도 4】



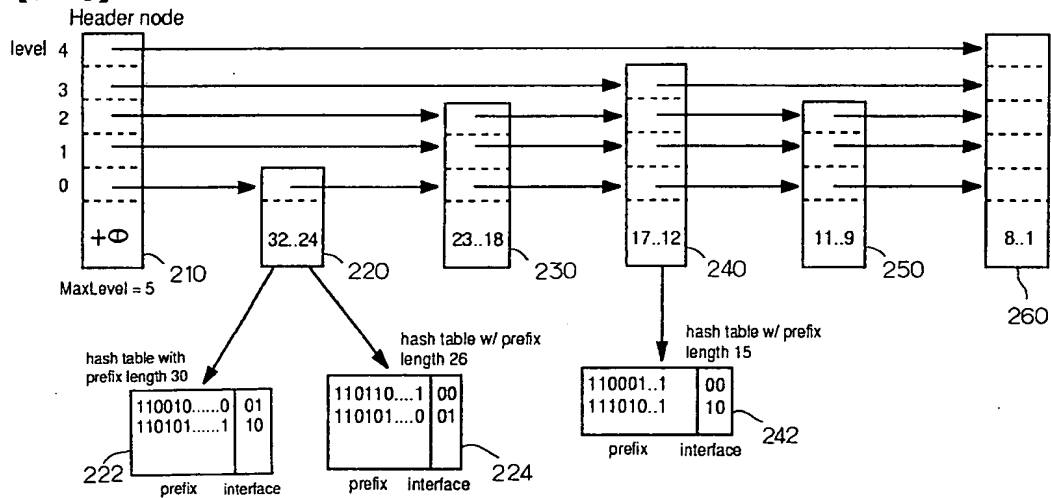
【도 5】

typical lookup
start

lookup start

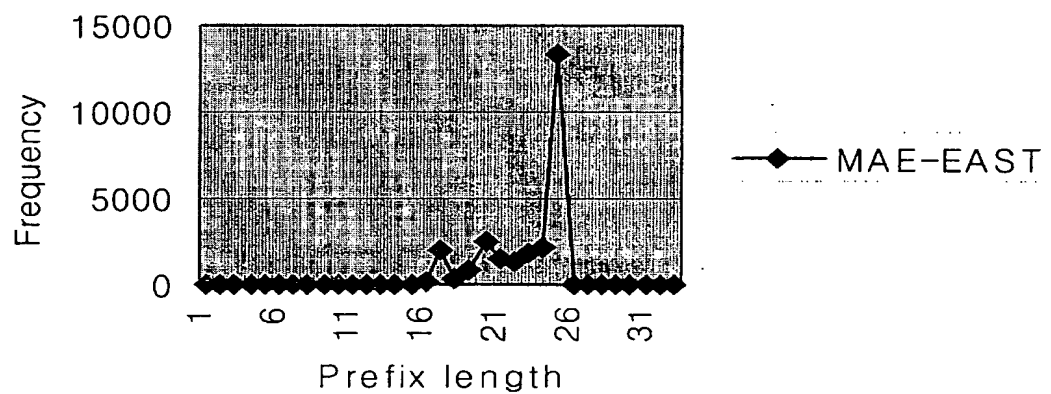


【도 6】



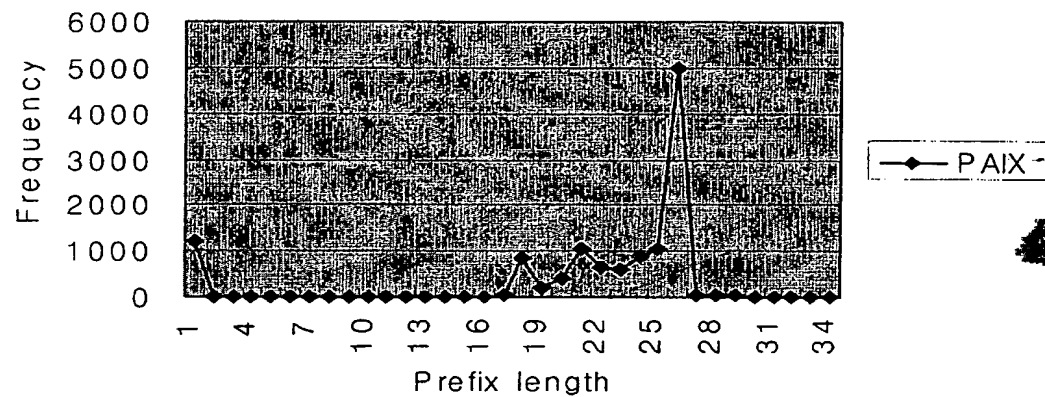
【도 7a】

Prefix length distribution

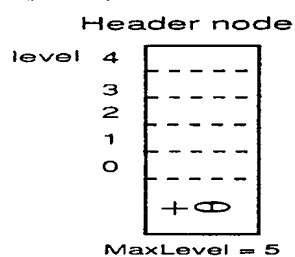


【도 7b】

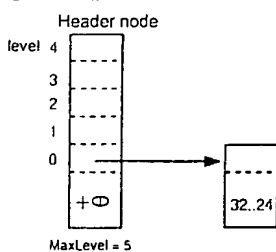
Prefix length distribution



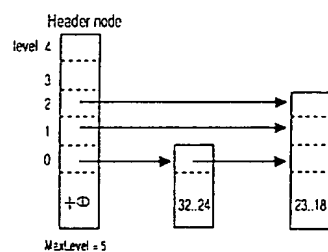
【도 8】



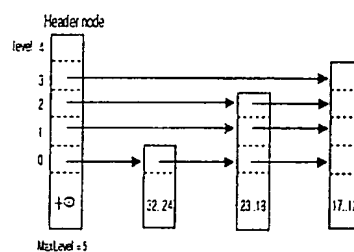
【도 9】



(a) 1st node

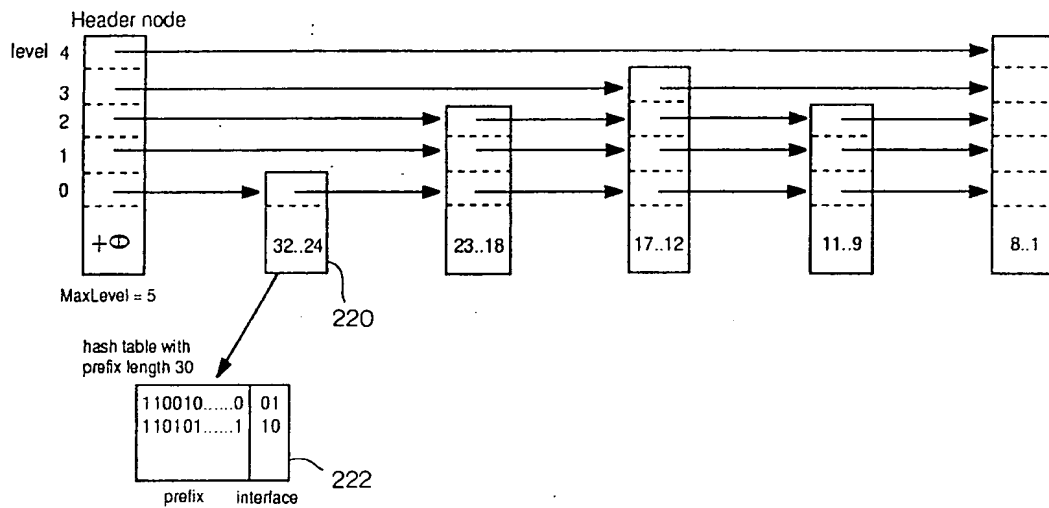


(b) 2nd node

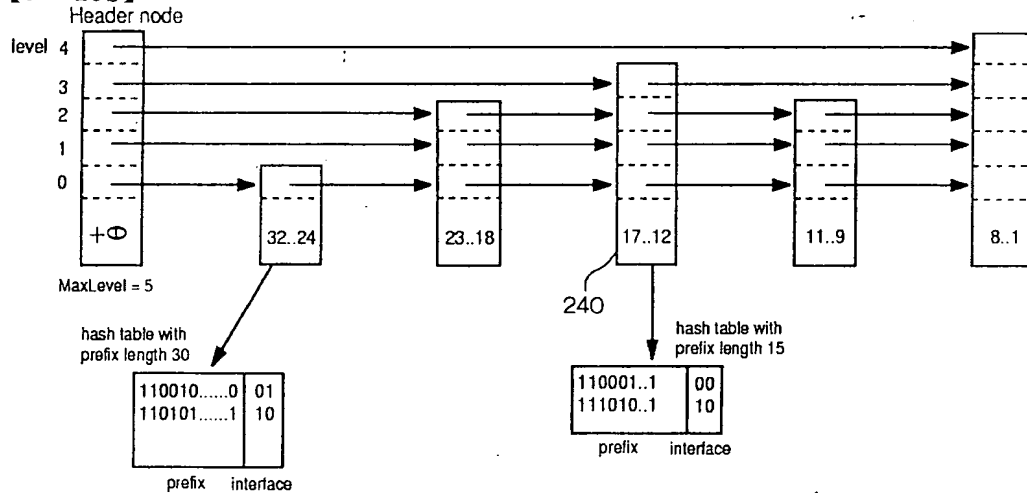


(c) 3rd node

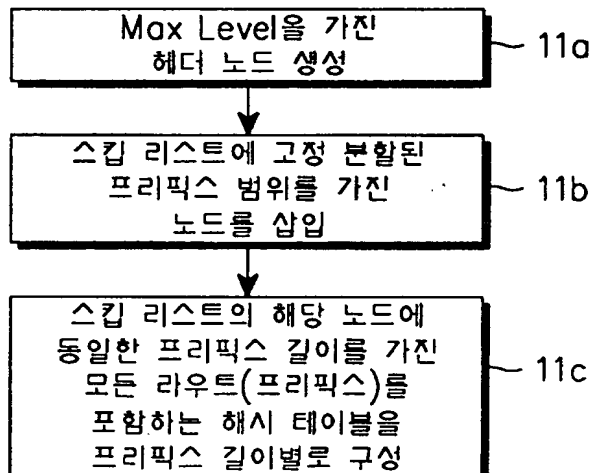
【도 10a】



【도 10b】



【도 11】



주어진 변경 라우트(프리픽스)에
대해 스킵 리스트에서 동일한
프리픽스 길이에 속하는 노드를 찾음

변경 리우트와 동일한 프리픽스
길이를 가진 해당 해시 테이블을
검색

```

graph TD
    In(( )) --> D{찾음?}
    D -- 아니오 --> Out1(( ))
    D -- 예 --> B[해당 해시 테이블 내의 발견된 라우트(프리픽스)를 변경 또는 삭제]
    B --> Out2(( ))
    style In fill:none,stroke:none
    style Out1 fill:none,stroke:none
    style Out2 fill:none,stroke:none
    
```

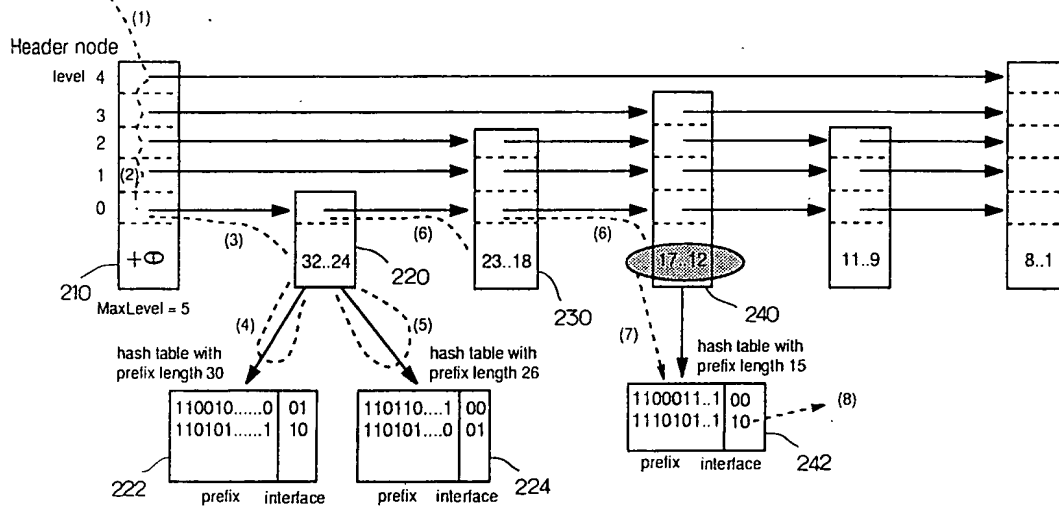
아니오 → 변경된 라우트를 위한
헤시 테이블 생성 12e

생성된 해시 테이블에
주어진 라우트를 삽입

```

graph TD
    15a[IP 헤더에서 주어진 목적지 어드레스에  
대해, 스킵 리스트에서 제1 노드를 찾음] --> 15b[방문한 노드의 상위 바운드 프리픽스  
범위로 부터 시작하는 각 프리픽스  
길이에 대해, 대응된 해시 테이블  
내의 프리픽스를 검색]
    15b --> 15c{해시 테이블 존재?}
    15c -- 예 --> 15d[주어진 목적지 어드레스와  
일치하는 해시 테이블  
내의 프리픽스를 찾음]
    15c -- 아니오 --> 15h[현재 프리픽스 길이 -1을 가진  
다음 해시 테이블로 이동]
    15d --> 15e{프리픽스 일치?}
    15e -- 예 --> 15f[일치한 프리픽스를  
최대 프리픽스로 리턴]
    15e -- 아니오 --> 15g[목적지 어드레스의  
비트 스트링을 1 감소]
    15g --> 15b
    15h --> 15i{해시 테이블 존재?}
    15i -- 예 --> 15c
    15i -- 아니오 --> 15j[스킵리스트에서 인접노드로  
이동]
    15j --> 15b
  
```

【도 16】



【도 17】

